

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**AN IMPROVED UNSUPERVISED MODELING
METHODOLOGY FOR DETECTING FRAUD IN VENDOR
PAYMENT TRANSACTIONS**

by

Gregory W. Rouillard

June 2003

Thesis Advisor:
Second Reader:

Samuel E. Buttrey
Lyn R. Whitaker

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2003	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: An Improved Unsupervised Modeling Methodology For Detecting Fraud In Vendor Payment Transactions			5. FUNDING NUMBERS	
6. AUTHOR(S) Gregory W. Rouillard				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Finance and Accounting Service, Internal Review Seaside (Operation Mongoose)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) In this thesis, we propose a standardized procedure for detecting fraud in Defense Finance and Accounting Service (DFAS) vendor payment transactions through Unsupervised Modeling (cluster analysis). Clementine Data Mining software is used to construct unsupervised models of vendor payment data using the K-Means, Two Step, and Kohonen algorithms. Cluster validation techniques are applied to select the most useful model of each type, which are then combined to select candidate records for physical examination by a DFAS auditor. Our unsupervised modeling technique utilizes all the available valid transaction data, much of which is not admitted under the current supervised modeling procedure. Our procedure standardizes and provides rigor to the existing unsupervised modeling methodology at DFAS. Additionally, we demonstrate a new clustering approach called Tree Clustering, which uses Classification and Regression Trees to cluster data with automatic variable selection and scaling. A Recommended SOP for Unsupervised Modeling, detailed explanation of all Clementine procedures, and implementation of the Tree Clustering algorithm are included as appendices.				
14. SUBJECT TERMS Cluster Analysis, Cluster Validation, Data Mining, Fraud Detection, K-Means, Kohonen, Tree Clustering, Two Step, Unsupervised Modeling.			15. NUMBER OF PAGES 173	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

**AN IMPROVED UNSUPERVISED MODELING METHODOLOGY FOR DETECTING
FRAUD IN VENDOR PAYMENT TRANSACTIONS**

Gregory W. Rouillard
Major, United States Marine Corps
B.S., United States Naval Academy, 1988

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

**NAVAL POSTGRADUATE SCHOOL
JUNE 2003**

Author: Gregory W. Rouillard

Approved by: Samuel E. Buttrey
Thesis Advisor

Lyn R. Whitaker
Second Reader

James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, we propose a standardized procedure for detecting fraud in Defense Finance and Accounting Service (DFAS) vendor payment transactions through Unsupervised Modeling (cluster analysis). Clementine Data Mining software is used to construct unsupervised models of vendor payment data using the K-Means, Two Step, and Kohonen algorithms. Cluster validation techniques are applied to select the most useful model of each type, which are then combined to select candidate records for physical examination by a DFAS auditor. Our unsupervised modeling technique utilizes all the available valid transaction data, much of which is not admitted under the current supervised modeling procedure. Our procedure standardizes and provides rigor to the existing unsupervised modeling methodology at DFAS. Additionally, we demonstrate a new clustering approach called Tree Clustering, which uses Classification and Regression Trees to cluster data with automatic variable selection and scaling. A standardized procedure for Unsupervised Modeling, detailed explanation of all Clementine procedures, and implementation of the Tree Clustering algorithm are included as appendices.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	OPERATION MONGOOSE AND VENDOR PAYMENT AUDITING	1
B.	A NEW CLUSTERING METHODOLOGY WITH AUTOMATIC VARIABLE SELECTION AND SCALING	3
C.	PURPOSE AND SCOPE OF THE THESIS	4
D.	OVERVIEW OF THESIS	4
II.	CLASSIFICATION AND DETECTION OF FRAUD	7
A.	CLASSIFYING FRAUD	7
B.	THE KNOWLEDGE BASE	8
C.	DETECTING FRAUD WITH SUPERVISED LEARNING	9
	1. Current Procedures	9
	2. Shortcomings of Supervised Modeling	10
	3. Potential Improvements With Unsupervised Modeling	11
III.	UNSUPERVISED LEARNING	13
A.	DEFINITION	13
B.	TYPES OF VARIABLES	13
	1. Interval-Scaled Variables	14
	2. Binary Variables	14
	a. <i>Definition of Symmetric and Asymmetric Binary Variables</i>	14
	b. <i>Measuring dissimilarity in binary variables</i>	15
	3. Nominal Variables	16
	4. Mixed Variable Types	17
C.	UNSUPERVISED LEARNING METHODS	19
	1. Partitioning	19
	2. Hierarchical Methods	20
	3. Self-Organizing Maps	22
D.	EVALUATION OF CLUSTERING RESULTS	24
	1. Optimum Number of Clusters for K-Means	24
	2. Cluster Validation	25
IV.	CLEMENTINE DATA MINING SOFTWARE	31
A.	OVERVIEW	31
B.	UNSUPERVISED LEARNING MODEL TYPES	31
	1. K-Means	31
	2. Two Step	35
	3. Kohonen	37
C.	SHORTCOMINGS OF CLEMENTINE UNSUPERVISED MODELING ..	40
V.	FINDING CLUSTERS IN NO2 VENDOR PAYMENT DATA USING CLEMENTINE UNSUPERVISED MODELS	43
A.	OVERVIEW OF THE PROCESS	43

B.	SOURCE DATA: THE NO2 POPULATION DATABASE	43
C.	DATA PRE-PROCESSING: THE BASIC FILTER & TYPE SUPERNODE	44
D.	K-MEANS MODELING: KMEANS_UNSUP_POP_GWR.STR	46
	1. Methodology	46
	2. All Fields: AB10 Models	49
	3. Numeric Fields Only: AB20 Models	51
	4. Numeric Fields Only (Principal Components Analysis): AB30 Models	52
E.	TWO STEP MODELING: TWOSTEP_UNSUP_POP_GWR.STR	53
F.	KOHONEN MODELING: KOHONEN_UNSUP_POP_GWR.STR	54
G.	MODEL ANALYSIS: ANALYSIS_UNSUP_POP_GWR.STR	57
	1. Overview	57
	2. K-Means and Two Step Models	58
	3. Kohonen Maps	58
	4. Implementation and Results	60
VI.	TREE CLUSTERING	63
A.	OVERVIEW	63
B.	CLASSIFICATION TREES	63
	1. Definition	63
	2. Construction	64
	3. Node Impurity and Deviance	65
C.	TREE CLUSTERING IMPLEMENTATION	66
D.	DEMONSTRATION OF THE TECHNIQUE	68
E.	APPLICATION TO VENDOR PAYMENT DATA	70
VII.	CONCLUSIONS AND RECOMMENDATIONS	73
A.	SUPERVISED LEARNING VS. UNSUPERVISED LEARNING	73
B.	RELATIVE COMPARISON OF K-MEANS, TWO STEP, AND KOHONEN CLEMENTINE MODELS	73
C.	RECOMMENDATIONS FOR INTERNAL REVIEW SEASIDE	75
D.	TREE CLUSTERING WITH LARGE DATA SETS	76
APPENDIX A.	NO2 POPULATION DATABASE	77
APPENDIX B.	NO2 POPULATION UNSUPERVISED MODELING IMPLEMENTATION AND DETAILED RESULTS	83
1.	BASIC FILTER & TYPE SUPERNODE	83
2.	KMEANS_UNSUP_POP_GWR MODELING STREAM	91
	a. Implementation	91
	b. Results	99
3.	TWOSTEP_UNSUP_POP_GWR MODELING STREAM	103
4.	KOHONEN_UNSUP_POP_GWR MODELING STREAM	105
5.	MODEL_ANALYSIS_POP_GWR ANALYSIS STREAM	107
	a. Implementation	107
	b. Results	115
APPENDIX C.	SPREADSHEET TOOLS FOR UNSUPERVISED MODELING	117

1.	SUM OF SQUARES	117
2.	CLUSTER CORRESPONDENCE ANALYSIS TEMPLATE	117
APPENDIX D. TREE CLUSTERING SPLUS IMPLEMENTATION		121
1.	S-PLUS IRIS DATA	121
2.	S-PLUS IMPLEMENTATION	122
a.	Function tree.clust()	122
b.	Application to Iris Noise Data	125
b.	Application to Vendor Payment Data	125
APPENDIX E. PROPOSED STANDARD OPERATING PROCEDURES FOR UNSUPERVISED MODELING TO DETECT FRAUD IN VENDOR PAYMENTS		127
1.	PURPOSE AND OVERVIEW	127
2.	DATA PRE-PROCESSING	128
a.	Source Data and SPSS Analysis	128
b.	The Basic Filter & Type Supernode	128
3.	MODEL BUILDING AND SELECTION	130
a.	K-Means Model Building	130
b.	Two Step Model Building	139
c.	Kohonen Model Building	139
4.	MODEL ANALYSIS AND RESULTS	140
LIST OF REFERENCES		147
INITIAL DISTRIBUTION LIST		149

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1	Generated Data for Clustering by Partitioning Methods	19
Figure 2	A Hierarchical Clustering Dendrogram of U.S. States from the S-PLUS [14] AGNES Clustering Algorithm	21
Figure 3	5x5 Kohonen Map of Generated Cluster Data	24
Figure 4	Evaluation of Optimum Number of Clusters	25
Figure 5	K-Means Node Model Options Dialog Box Model Tab ...	32
Figure 6	K-Means Node Model Options Dialog Box Expert Tab (Default Values)	33
Figure 7	K-Means Generated Models Dialog Box Model Tab	35
Figure 8	Two Step Node Model Options Dialog Box Model Tab ..	36
Figure 9	Two Step Generated Models Dialog Box Model Tab	37
Figure 10	Kohonen Node Model Options Dialog Box Model Tab ...	38
Figure 11	Kohonen Node Model Options Dialog Box Expert Tab ..	39
Figure 12	Kohonen Generated Models Dialog Box Summary Tab ...	40
Figure 13	Basic Filter & Type Supernode	45
Figure 14	Kmeans_unsup_pop_GWR Main Palette	47
Figure 15	Model Building Script	48
Figure 16	K-Means Models Built With All Fields	50
Figure 17	K-Means Numeric Only Models	52
Figure 18	K-Means PCA Numeric Only Models	53
Figure 19	TwoStep_unsup_pop_GWR Main Palette	54
Figure 20	Kohonen_unsup_GWR Main Palette	55
Figure 21	5x5 Kohonen Map	56
Figure 22	10x11 Kohonen Map	56
Figure 23	Analysis_unsup_pop_GWR Main Palette	57
Figure 24	Classification Tree of the S-PLUS Iris Data	64
Figure 25	Classification Tree Illustrating Degrees of Dissimilarity	67
Figure 26	Comparison of Records Selected by Supervised and Unsupervised Models	73
Figure 27	SQL Node Dialog Box for N02_STA_POP2000 Database ..	81
Figure 28	Basic Filter Node Dialog Box	83
Figure 29	To String Filler Node Dialog Box	84
Figure 30	Basic Type Node Dialog Box	85
Figure 31	Distributions and Statistics Supernode	86
Figure 32	PMT_METH Distribution Node Dialog Box	87
Figure 33	PMT_METH Distribution Plot	87
Figure 34	Numeric Statistics Node Dialog Box	88
Figure 35	Numeric Statistics Node Output	89
Figure 36	ValSet Derive Node Dialog Box	90
Figure 37	Contract Derive Node Dialog Box	91

Figure 38	K-Means Model Node Dialog Box	92
Figure 39	PCA Model Node Dialog Box	92
Figure 40	AB10 Models Supernode	93
Figure 41	ValSet Select Node (currently selects ValSet "B") ..	94
Figure 42	A/B Validation Models Supernode	94
Figure 43	K-Means A30/B30 Matrix Dialog Box	95
Figure 44	Sum of Squares Supernode	96
Figure 45	_Square Derive Node Dialog Box	96
Figure 46	Within-Cluster Sum of Squares Set Globals Dialog Box	97
Figure 47	_Sum_Square Derive Node Dialog Box	98
Figure 48	Sum of Squares Aggregate Node Dialog Box	98
Figure 49	K-Means06AB20 Generated Model Node, Summary Tab ..	100
Figure 50	K-Means06AB20 Generated Model Node, Model Tab	101
Figure 51	K-Means06AB50 Generated Model Node, Model Tab	102
Figure 52	K-Means06AB60 Generated Model Node, Model Tab	102
Figure 53	K-Means06AB70 Generated Model Node, Model Tab	103
Figure 54	TwoStep Model Node Dialog Box	104
Figure 55	TwoStep07AutoAB20 Generated Models Dialog Box, Model Tab	104
Figure 56	Kohonen Model Node Dialog Box, Model Tab	105
Figure 57	Kohonen Model Node Dialog Box, Expert Tab	106
Figure 58	Kohonen Model Plot Dialog Box	106
Figure 59	Kohonen Generated Model Dialog Box, Summary Tab ..	107
Figure 60	Sparse Prototypes Supernode	108
Figure 61	Aggregate Node Settings	108
Figure 62	Table of Kohonen Prototypes Sorted in Descending Order by Number of Transactions	109
Figure 63	KSOM_10 Derive Node Settings	110
Figure 64	Contract Count Supernode	111
Figure 65	Merge Node Dialog Box, Merge Tab	111
Figure 66	Merge Node Dialog Box, Filter Tab	112
Figure 67	Orphans Supernode	113
Figure 68	Merge Node Filter Settings	113
Figure 69	TS_Orphan Derive Node Settings	114
Figure 70	Triple Orphans Select Node Settings	114
Figure 71	Distribution of Orphan Transactions by K-Means Cluster	115
Figure 72	Distribution of Orphan Transactions by Two Step Cluster	115
Figure 73	Sum of Squares Spreadsheet Tool	117
Figure 74	Cluster Correspondence Analysis Template Analysis Worksheet	118
Figure 75	Cluster Correspondence Analysis Template 6 Clusters Worksheet	119
Figure 76	Basic Filter & Type Supernode	128

Figure 77	Distributions and Statistics Supernode	129
Figure 78	Kmeans_N02pop Stream	131
Figure 79	ValSet Select Node	132
Figure 80	K-Means Model Building Script (Script Tab of the Stream Properties Dialog Box)	133
Figure 81	Generated K-Means Models	133
Figure 82	Sum of Squares Supernode	134
Figure 83	_Square Derive Node Dialog Box	135
Figure 84	Within-Cluster Sum of Squares Set Globals Dialog Box	135
Figure 85	_Sum_Square Derive Node Dialog Box	136
Figure 86	Sum of Squares Aggregate Node Dialog Box	136
Figure 87	Example of Sum of Squares Plot	137
Figure 88	Matrix Node Settings Tab	138
Figure 89	Kohonen Model Node Expert Tab	140
Figure 90	Model_analysis_N02pop	141
Figure 91	Contract Count Supernode	142
Figure 92	Merge Node Dialog Box, Merge Tab	142
Figure 93	Merge Node Dialog Box, Filter Tab	143
Figure 94	Orphans Supernode	144
Figure 95	Merge Node Filter Settings	144
Figure 96	TS_Orphan Derive Node Settings	145
Figure 97	Triple Orphans Select Node Settings	146

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1	Binary Variable Contingency Table (from [12])	15
Table 2	A/B Validation Contingency Table for Generated Cluster Data	27
Table 3	A/B Validation Contingency Table for Generated Cluster Data (Rearranged to Illustrate Cluster Mapping on Main Diagonal)	27
Table 4	Example of $r \times c$ Contingency Table	28
Table 5	Effect of Reordering Data on K-Means Models Built With Categorical Data	51
Table 6	Sparse Prototype Transaction Counts	59
Table 7	Orphan Transaction Distribution	61
Table 8	Contingency Table for Tree Clustering Scaled Iris Noise Data	69
Table 9	Contingency Table for Clustering Scaled Iris Noise Data with PAM (Standardized Variables)	70
Table 10	Contingency Table for Tree Clustering Knowledge Base	71
Table 11	Comparison of Unsupervised Model Types	74
Table 12	Modified Fields To Use Matrix (on four pages)	77
Table 13	PCA Factor Analysis Component Matrix	99
Table 14	A/B Validation Matrix for K-Means06AB20	101
Table 15	Cross-Tabulation of Cluster Assignment, K-Means06AB20 vs. K-Means06AB50 Models	102
Table 16	Cross-Tabulation of Cluster Assignment, K-Means06AB60 vs. K-Means06AB70 Models	103
Table 17	A/B Validation Matrix for TwoStep07AutoAB20	105
Table 18	Example of Original Iris Data	121
Table 19	Example of Scaled Iris Data With Noise Variables	121

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and assistance of many people. It is impossible to thank everyone who has contributed, but I would like to recognize the contributions of those who helped the most.

LTC Chris Nelson, Dave Riney, and Randy Faulkner from Internal Review Seaside provided indispensable assistance in everything from teaching me the basics of fraud detection and data mining to helping me learn to use Clementine effectively. Thanks also to all the other personnel at IR Seaside for their unfailing courtesy and assistance.

Professor Sam Buttrey is the local guru of S-PLUS and has been a great source of guidance in my journey into the world of unsupervised modeling. His patience and help with the Tree Clustering concept and execution were admirable.

Professor Robert Koyak suggested using Cramer's Coefficient as a tool for cluster model validation, a valuable piece of advice.

Finally, my heartfelt and sincerest thanks to my beloved family, whose love, patience, and support are the foundation of all my endeavors.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The Defense Finance and Accounting Service (DFAS) Internal Review Seaside (IR Seaside) office, also known as Operation Mongoose, is responsible for identifying potentially fraudulent transactions in vendor payment data. Their primary tool is data mining of vendor payment data to identify candidate transactions for manual audit by DFAS accountants. The current procedure relies heavily on supervised methods such as Classification and Regression Trees and Neural Networks, which predict the fraud classification of transactions in an audit population. These supervised models are "trained" using a Knowledge Base of transactions from 17 proven fraud cases. Unfortunately, this data is outdated and incomplete, so supervised models built with the Knowledge Base may not effectively exploit all the characteristics of audit population data.

Unsupervised modeling, or cluster analysis, is a data mining technique that finds patterns or groupings in data without the need for a response variable (such as fraud classification). Unsupervised models are specific to a particular data set, and independent of any external data for model construction. The current unsupervised modeling process is neither rigorous nor standardized. Of the total number of transactions selected for manual audit, supervised modeling is used to identify 80%, unsupervised modeling accounts for 10%, and the remaining 10% are selected at random. Supervised and unsupervised models are trained using SPSS, Inc.'s data mining software Clementine, Version 7.0.

The intent of this thesis is to develop a standardized, rigorous unsupervised modeling methodology that utilizes all available valid transaction data and analyzes audit population transactions independent of the Knowledge Base. Clementine's K-Means, Two Step, and Kohonen algorithms are used to construct unsupervised models of audit population payment data, and then cluster validation techniques are applied to select the most useful model of each type. Finally, these three models are combined to select candidate records for physical examination by a DFAS auditor.

The selection of candidate records for audit is based on the assumption that all the transactions belonging to the same contract are somehow similar, and should be grouped together. After clustering the data, any transaction that does not fall within the "home" cluster of its parent contract is considered an "orphan." Transactions that are identified as orphans under all three clustering schemes are selected for audit.

This methodology is not intended to replace the current system of supervised modeling; rather it should be considered complementary. It is desirable to identify different candidate transactions with each of the two methods, producing a more robust collection of transactions for manual audit.

This improved methodology was developed using a previously audited population of vendor payment transactions from the US Navy STARS system in Norfolk, VA. A total of 155 transactions (out of over 198,000) were identified as orphans by all three of the unsupervised

models. The previously conducted supervised modeling effort identified 243 potentially fraudulent transactions in the Norfolk data; there were only two transactions selected by both methods, illustrating the independence and complementary nature of the two techniques.

Deliverables to IR Seaside include the Clementine files used to develop the methodology, a Proposed Standard Operating Procedure for Unsupervised Modeling, two spreadsheet tools for cluster validation, and a two-hour training presentation for all Operation Mongoose personnel.

This thesis additionally demonstrates a new clustering approach called Tree Clustering, which uses Classification and Regression Trees to cluster data with automatic variable selection and scaling. This technique is successfully demonstrated on a small set of simple data using Insightful Corporation's SPLUS statistics and data analysis software. The technique is also applied to the DFAS Knowledge Base, with mixed results.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OPERATION MONGOOSE AND VENDOR PAYMENT AUDITING

The Defense Finance and Accounting Service (DFAS) is responsible for disbursing nearly all the funds expended by the Department of Defense. Given the enormous number of taxpayer dollars that are paid in the services' Vendor Payment systems, fraud is a major concern. In the mid-1990's, the Office of the Secretary of Defense (OSD) sponsored a project to uncover fraudulent vendor payment transactions called "Operation Mongoose." This project was subsequently undertaken by the DFAS Internal Review section in Seaside, CA (IR Seaside). To identify fraudulent payments, a DFAS examiner reviews the documentation on hundreds of vendor payments, selected out of the hundreds of thousands of total transactions. An effective and efficient selection process is critical: auditing is very time-consuming, there are a limited number of examiners, and fraudulent payments are very rare in proportion to the total number of transactions. Data mining was selected as the principal tool to select candidate records for audit.

IR Seaside contracted Dr. Dean Abbott of Abbott Consulting, Inc., to develop its data mining methodology. Dr. Abbott *et al.* devised a data mining process [6] using the popular data mining software Clementine [4]. Their procedure combines a Knowledge Base (KB) of known (successfully prosecuted) fraud transactions and samples of transactions from the population being examined (presumably not fraudulent) to "train" various classification tree, rule-based, and neural network models to detect fraudulent

payments. This technique is called "supervised modeling." The supervised models are evaluated, compared, and combined in a weighted voting scheme, which results in the selection of candidate records for audit (transactions that the models predict are likely to be fraudulent). Under the current DFAS system, the majority of transactions (80%) selected for manual audit come from the combined supervised model (or are related to those selected), with the remainder selected randomly (10%) or through unsupervised modeling (10%).

Since March 2000, the IR Seaside team has conducted audits at thirteen vendor payment system sites using the methodology described above. Although many of the payments audited after having been selected by data mining have one or more Conditions Needing Improvement (CNI; some deficiency or error which might indicate potential fraud), there has only been one case discovered with evidence of fraud strong enough to warrant prosecution.

The Knowledge Base (KB) of fraudulent transactions that is used to "train" the supervised models used in the current data mining process is small and outdated. There are many data fields in the populations to be investigated that are not populated in the KB, and thus are not used in the current supervised modeling process. These fields contain information which if included could presumably enhance the detection of fraudulent payments. Although Dr. Abbott *et al.* used unsupervised learning (cluster analysis) in their initial classification of the fraud transactions in the KB, the unsupervised modeling conducted in the current data mining process is neither rigorous nor

standardized. Unsupervised learning should be used more extensively to exploit the many data fields that are not populated in the KB, which thus go unused in supervised modeling. Exploration of this otherwise “wasted information” could potentially enhance the detection of data patterns that might indicate fraudulent activity.

B. A NEW CLUSTERING METHODOLOGY WITH AUTOMATIC VARIABLE SELECTION AND SCALING

When finding clusters in data, the choice of variables included for modeling can have an impact on the results. There may be one or more fields whose values are unrelated to anything of interest, whose inclusion as variables for modeling might produce incorrect or misleading results. A useful clustering methodology must be able to detect and eliminate such “noise” variables.

Another challenge when clustering data with continuous variables is the choice of scale. Different scale choices can produce wildly different, and possibly misleading, clustering results. Thus, an effective means of scaling is desirable for a clustering methodology, as discussed in [12].

Buttrey has proposed a new method called “Tree Clustering” in [2]. This technique uses a set of regression or classification trees (one for each original variable) to find similarities among observations (observations which tend to fall into the same leaves being similar). This approach automatically selects the most important variables for clustering and is scale-

independent. The Tree Clustering method is described in detail in Chapter VI.

C. PURPOSE AND SCOPE OF THE THESIS

The purpose of this thesis is twofold: first, to develop a useful, rigorous, standardized cluster analysis methodology for IR Seaside using the Clementine data mining software; second, to demonstrate the tree clustering methodology on vendor payment data.

This thesis will be limited in scope to analyzing DoD vendor payment data using unsupervised modeling (cluster analysis). It will not address any issues involving supervised modeling other than to point out shortcomings of the current procedures. The specific data used to develop the unsupervised modeling methodology is the N02_STA_POP_2000 database of US Navy STARS transactions conducted in Norfolk, VA, from October 2000 to March 2002.

D. OVERVIEW OF THESIS

This thesis is organized into four general areas: background information, implementation and results, conclusions and recommendations, and appendices.

Chapters II, III, and IV contain the background information from which the methodology is developed. Chapter II describes the current state of classification and detection of fraud in vendor pay transactions, the Knowledge Base and supervised modeling, and potential improvements available with unsupervised modeling. Chapter III is a primer on the basics of unsupervised learning,

including data types, modeling methods, and validation methods. Chapter IV presents an introduction to the Clementine Data Mining software's basic terminology, functions, and unsupervised model types.

Chapter V presents the implementation and results of unsupervised modeling on the Norfolk vendor pay data. Each of the four Clementine streams is discussed in detail, as well as the final clustering results. Chapter VI contains a thorough discussion of the Tree Clustering methodology's theory, implementation, and results.

Chapter VII presents the conclusions drawn from analysis of the results obtained from Clementine unsupervised modeling and application of the Tree Clustering algorithm. Appendix A displays a detailed description of the four Clementine streams discussed in Chapter V, as well as supporting results. Appendix B is a proposed Standard Operating Procedure (SOP) for Unsupervised Modeling for Operation Mongoose. Appendix C contains the code for S-PLUS implementation of the Tree Clustering methodology.

THIS PAGE INTENTIONALLY LEFT BLANK

II. CLASSIFICATION AND DETECTION OF FRAUD

A. CLASSIFYING FRAUD

Dr. Dean Abbott *et al.* extensively examined the Knowledge Base of historical fraud cases and consulted at length with the accounting experts in DFAS to develop a classification scheme for vendor pay fraud. This process is exhaustively detailed in Abbott's Final Report [6]. The end result was four classes of fraud: Big Systematic, Small Systematic, Piggyback, and Opportunistic.

Big Systematic and Small Systematic fraud are characterized by a long-term process of well planned actions designed to defraud the government. The primary difference is one of scale of money stolen. Piggyback fraud occurs when the criminal "piggybacks" a fraudulent payment onto other, legitimate ones. Finally, opportunistic fraud is just what it seems: a relatively small-scale theft of opportunity.

The class of fraud assigned to a transaction is used as the output or response variable in supervised modeling. When a vendor pay site's population of transactions is used as training, testing, and validation data to build supervised models, each transaction is assigned the class of "NF" for "not fraud," on the assumption that all transactions in the population are legitimate. Thus the combined supervised models used by Operation Mongoose are designed to select potentially fraudulent payments based on the fraud class predicted by the model trained on the KB data.

B. THE KNOWLEDGE BASE

Operation Mongoose's Data Mining Knowledge Base (KB) is an historical repository of 17 successfully prosecuted fraud cases consisting of 442 total transactions, conducted from February 1989 to June 1997. Each case is classified according to one of the four classes of fraud described in the previous paragraph. Each transaction contains 59 fields of original, transformed, and derived data. The KB has several shortcomings, which brings into question its utility in predicting future fraud cases: first, it is outdated; second, many of the fields found in the populations are not populated in the KB; and finally, there is missing data.

The age of the KB is problematic for two reasons. First and foremost, all of the KB transactions were conducted before the advent of electronic payments, so the characteristics of these transactions can be expected to differ substantially from current EFT-type payments. This contributes to the problem illustrated in the next paragraph as well. Second, the fraudulent payments in the KB represent the "state of fraud" at the time. It is not reasonable to presume that fraudulent practices have not evolved over time; presumably modern fraudsters would use different methods from their predecessors.

The problem of unpopulated fields in the KB relative to the populations being examined relates to the age of the Knowledge Base, the evolution of data collection practices, and the different types of vendor pay accounting systems in use today. As mentioned in the previous paragraph, one of the most significant deficiencies of the KB is the lack of

any electronic payment information. Of course this is unavoidable given the age of the KB, but it greatly reduces the utility of models built using KB data to predict fraud in populations where electronic payments exist. Besides EFT information, there are other data that is captured today whose capture was infeasible prior to the advent of modern computers and relational databases. These types of fields are unpopulated in the KB, of course. Finally, there are four different types of vendor payment systems in use in the DOD today, each of which has unique fields for data entry as well as more common ones. These unique fields are not populated in the Knowledge Base.

Of the 26,078 possible data entries in the KB, 596 of them are missing, primarily in two fields. Monteiro in [13] conducted an analysis of the Knowledge Base and concluded that the pattern of missing values is nonrandom. This nonrandom pattern results in conditional dependence among the four fraud classes, increasing the likelihood of misclassification. Combined with the fact that current business practices may differ among audit sites, it is possible that supervised models trained on the KB are predicting either the wrong type of fraud, or predicting something other than fraud altogether. These concerns highlight the need to update, expand, and improve the KB for successful supervised modeling.

C. DETECTING FRAUD WITH SUPERVISED LEARNING

1. Current Procedures

The current DFAS Standard Operating Procedure for Data Mining [7] is extensively detailed in its discussion of

supervised modeling. The process begins with random division of the population into eight "splits," each of which is further subdivided into training, testing, and validation sets. Transactions from the KB are then assigned to each of the 24 sets in a sequential, orderly manner, resulting in eight sets of Training, Testing, and Validation data containing both known fraud cases and records from the population being examined.

Next, several different modelers independently build a model (or set of models) on a different split or set of splits, using Clementine supervised models such as Classification & Regression Trees, C5.0 Decision Trees, and Neural Networks. The "best" of these models (in terms of correctly predicting the fraud class of the KB transactions in their data splits) are combined in a complex weighted voting scheme, which iteratively produces a list of candidate records for further investigation. These candidate records and all related records from the population are then selected for manual audit. [7]

2. Shortcomings of Supervised Modeling

The primary shortcoming of the supervised modeling methodology currently in place is its reliance on the outdated, incomplete, and potentially misclassified Knowledge Base, as detailed in Section B of this Chapter. Additionally, the supervised modelers at Operation Mongoose work very hard to create complex models and combinations of models that consistently "nail" all the KB transactions of a particular type, which is overfitting the data.

Although the population data is randomly divided, the assignment of KB transactions to the data splits is

predetermined, not random, which brings into question the validity of the predictions made by the resulting models. Finally, the supervised models do not use many of the data fields that are available in the population because they are not populated in the KB.

3. Potential Improvements With Unsupervised Modeling

The primary potential improvement with unsupervised modeling is the ability to exploit all the data in the population without regard to the Knowledge Base. Additionally, an unsupervised model will reveal actual patterns in the population data, independent of the preconceived (and potentially incorrect) fraud classifications in the KB. There are, of course, deficiencies and challenges associated with unsupervised learning; these are addressed in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

III. UNSUPERVISED LEARNING

A. DEFINITION

Unsupervised learning, also known as cluster analysis or data segmentation, can be defined as the field of statistical modeling that does not predict the value of a response variable as a function of one or more factors. Rather, an unsupervised model is used to describe a data set in its entirety, grouping together similar observations into distinct clusters. The "distance" between clusters depends on their degree of dissimilarity; observations that fall into two clusters that are "close together" are more similar to each other than observations from clusters that are "far apart."

Some measure of the similarity between observations must be calculated in order to find clusters in the data set. Most clustering algorithms utilize a numeric matrix (called a similarity or dissimilarity matrix) to represent the distances between observations. Thus any non-numeric variables must be coded numerically in terms of similarity or dissimilarity. For consistency, I will discuss similarity between observations in terms of distance or dissimilarity.

B. TYPES OF VARIABLES

The measure of similarity between observations depends primarily on the type of data that makes up the observation. I will consider only the three data types found in the vendor payment data: interval-scaled, binary, and nominal variables, as well as mixed variables. This

discussion of variable types and dissimilarity measures follows Kauffman and Rousseau in [12]. Note that there are other dissimilarity measures possible than those described in the following sections.

1. Interval-Scaled Variables

An interval-scaled variable takes on negative or positive real values on a linear scale. The most common measure of dissimilarity, or distance, for this data type is Euclidean distance. For a pair of observations i and j with p interval-scaled variables per observation, denoted by $x_{i1}, \dots, x_{ip}$ and $x_{j1}, \dots, x_{jp}$ respectively, the distance

(dissimilarity) is $d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ip} - x_{jp})^2}$.

2. Binary Variables

A binary variable takes on only one of two values or states, such as one and zero, on and off, or true and false. In data applications, binary variables are usually coded using one and zero. There are two types of binary variables, symmetric and asymmetric.

a. Definition of Symmetric and Asymmetric Binary Variables

A symmetric variable, the most common type, is one where each state is equally informative, and it does not matter which state is coded as a one. For example, the variable "sex" has possible states "male" and "female." It can be stated with confidence that two observations which are both "female" both have the same sex. An asymmetric binary variable, however, possesses states that are not equally informative, such as the "presence or absence of a

relatively rare attribute.”[12] The convention is to code the most important, or rarest, outcome, with a one. For example, consider the variable “hair color” with states “red” and “not red.” In this case, two observations with “hair color” of “not red” cannot reasonably be assumed to have the same color. Asymmetric binary variables are not as common as symmetric binary variables.

b. Measuring dissimilarity in binary variables

Consider two observations i and j , each consisting of p binary variables. The first step in calculating their dissimilarity is to consider a 2-by-2 contingency table for them, such as shown in Table 1. In this table, a is the number of data elements (binary variables) that equals one for both observations, b and c represent the number of variables that are different between the two observations, and d is the number of data elements that equals zero for both observations. The sum $a+b+c+d$ equals the total number of variables, p . The case where both observations have ones (a in Table 1) is also called a positive match, whereas observing two zeros (d in Table 1) is likewise called a negative match.

		observation j		
		1	0	
observation i	1	a	b	$a+b$
	0	c	d	$c+d$
		$a+c$	$b+d$	p

Table 1 Binary Variable Contingency Table (from [12])

The distinction between the two kinds of binary variables is important when considering the dissimilarity measure to be used for clustering in a particular application. When considering symmetric binary variables, positive and negative matches are equally important, so *invariant* dissimilarity coefficients, in which a and d carry equal weight, are appropriate. The most common invariant coefficient (which is also the simplest and most intuitive) is called the *simple matching coefficient*, and is also known as the *M-coefficient* or *affinity index*. It is defined as the proportion of disagreements between the two observations i and j :

$$d(i, j) = \frac{b+c}{a+b+c+d}.$$

When considering an asymmetric binary variable, however, the most important (and rarest) outcome is typically coded as a one, so a positive match is more significant than a negative match. Thus a *noninvariant* coefficient is required, one that gives more weight to a than d . The most popular noninvariant coefficient, the Jaccard coefficient, looks remarkably similar to the simple matching coefficient except that d is left out of the equation entirely:

$$d(i, j) = \frac{b+c}{a+b+c}.$$

3. Nominal Variables

A nominal variable is one that takes on one of a finite set of values, such as a field containing hair color, with possible values brown, blond, black, red, and other. Generally speaking, these states or values are

coded as integers $1, 2, 3, \dots, M$, where M is of course the total number of possible values, and each integer corresponds to one of the actual values (brown = 1, blond = 2, etc.). These states are unordered, and each one is equally important, so the coding can be done in any order.

Returning to the example of observations i and j , we now consider each one to consist of p nominal variables. The most common measure of dissimilarity between them is the simple matching approach:

$$d(i, j) = \frac{p - u}{p},$$

where u is the total number of matches (the number of variables out of p that have the same value for both observations). Because the coding of possible states is unordered, this dissimilarity measure is invariant.

4. Mixed Variable Types

In the event that all of the variables in a data set are of the same type (interval scaled, binary, or nominal), a dissimilarity matrix can be constructed using the dissimilarity measures described in the previous three subsections. However, in many real-world data sets, there are variables of more than one type. Therefore, to cluster mixed variable type observations, some combined dissimilarity measure must be used.

Kaufman and Rousseeuw in [12] describe a generalization of the method of Gower [9], which applies to all of the variable types previously discussed. Consider a data set of n observations each with p variables of mixed types, with the following definitions:

➤ x_{if} is the f^{th} variable of observation i

➤ $\delta_{ij}^{(f)}$ is an indicator variable

$$\begin{aligned}\delta_{ij}^{(f)} &= 1 \text{ if both } x_{if} \text{ and } x_{jf} \text{ are nonmissing} \\ &= 0 \text{ if either } x_{if} \text{ or } x_{jf} \text{ is missing or if} \\ &\text{variable } f \text{ is an asymmetric binary variable and} \\ &\text{observations } i \text{ and } j \text{ constitute a 0-0 match}\end{aligned}$$

➤ R_f is the range of an interval-scaled variable f

➤ $d_{ij}^{(f)}$ is the contribution of the f^{th} variable to the dissimilarity between observations i and j

o for a binary or nominal variable f

$$\begin{aligned}d_{ij}^{(f)} &= 1 \text{ if } x_{if} \neq x_{jf} \\ &= 0 \text{ otherwise}\end{aligned}$$

o for an interval-scaled variable f

$$d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{R_f}; \quad 0 \leq d_{ij}^{(f)} \leq 1$$

Note that Euclidean distance is not used in this case.

Using the preceding definitions, the overall dissimilarity between observations i and j is defined as:

$$d(i, j) = \frac{\sum_{f=1}^p \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^p \delta_{ij}^{(f)}}$$

Because $\delta_{ij}^{(f)} \in \{0,1\}$ and $0 \leq d_{ij}^{(f)} \leq 1$, $0 \leq d(i, j) \leq 1$ and it can be entered directly into an $n \times n$ dissimilarity matrix for use in a clustering algorithm.

C. UNSUPERVISED LEARNING METHODS

1. Partitioning

A partitioning method groups a data set of n observations into k distinct clusters. This grouping must satisfy the requirements of a partition: each group must contain at least one observation, and each observation must fall in exactly one group. The user must specify the value of k before commencing clustering. A partitioning algorithm can construct any specified number of clusters, but not all such groupings will be natural or useful for the given data. Therefore final selection of k is dependent on trial and error, expert opinion, or other methods. This problem is discussed more fully in section D of this chapter. Figure 1 shows a data set generated to illustrate partitioning clustering.

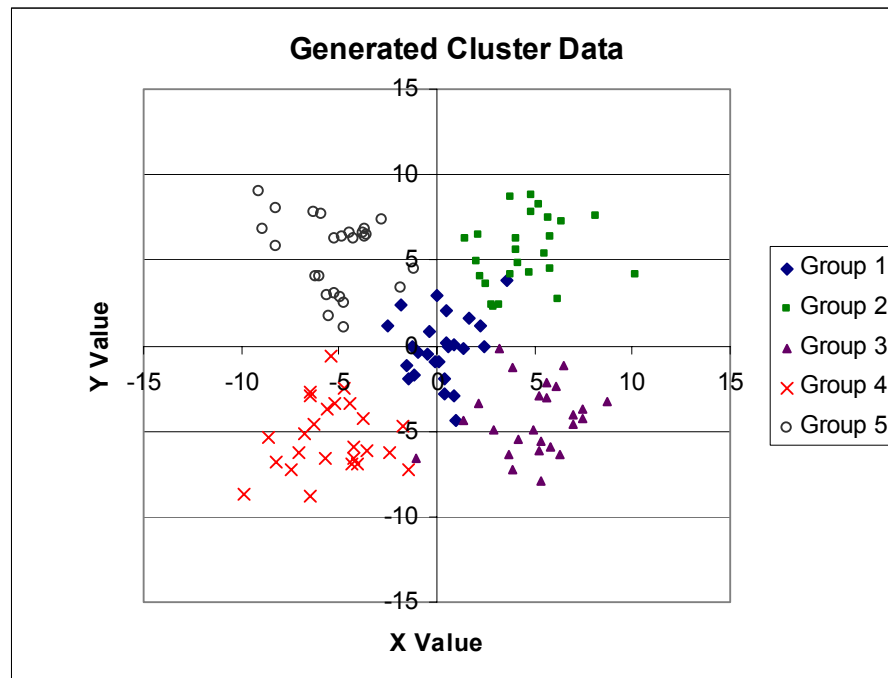


Figure 1 Generated Data for Clustering by Partitioning Methods

A popular partitioning method for data clustering discussed in [12] is called K-Means. This technique uses Euclidean distance between observations and cluster centers as its dissimilarity measure, so it is most applicable to interval-scaled data. Some software (including Clementine) transforms categorical data for clustering by this method; see Chapter IV, Section B for a detailed discussion.

The basic K-Means algorithm uses three steps to cluster data:

1. During an initial pass through the data, k initial cluster centers are selected.
2. In the second pass, the Euclidean distance from each observation to the nearest center is calculated, and the observation is initially assigned to that cluster.
3. During the third data pass, the cluster centers are updated based on the mean distance between all observations within it.

Steps two and three are iterated until the decrease in mean distance achieved by changing the cluster assignment of any observation is below some specified threshold, or a specified maximum number of iterations is reached. Each observation is ultimately assigned a cluster number label and (Euclidean) distance from its cluster center.

2. Hierarchical Methods

While a partitioning method seeks to create a predetermined number of groups of observations, hierarchical clustering results in every possible number of clusters from 1 (all observations in the same cluster) to n (one observation per cluster). As the name implies, as k increases from 1 to n , clusters on each “level” of the hierarchy consist of subsets of the clusters on the level above (smaller k). For example, if k increases from five to six, the sixth cluster is a subset of one of the other

five; conversely, if k decreases from four to three, one of those three clusters will contain all the observations found in some pair of the original four clusters. This type of clustering is best visualized as a dendrogram or tree (see Figure 2).

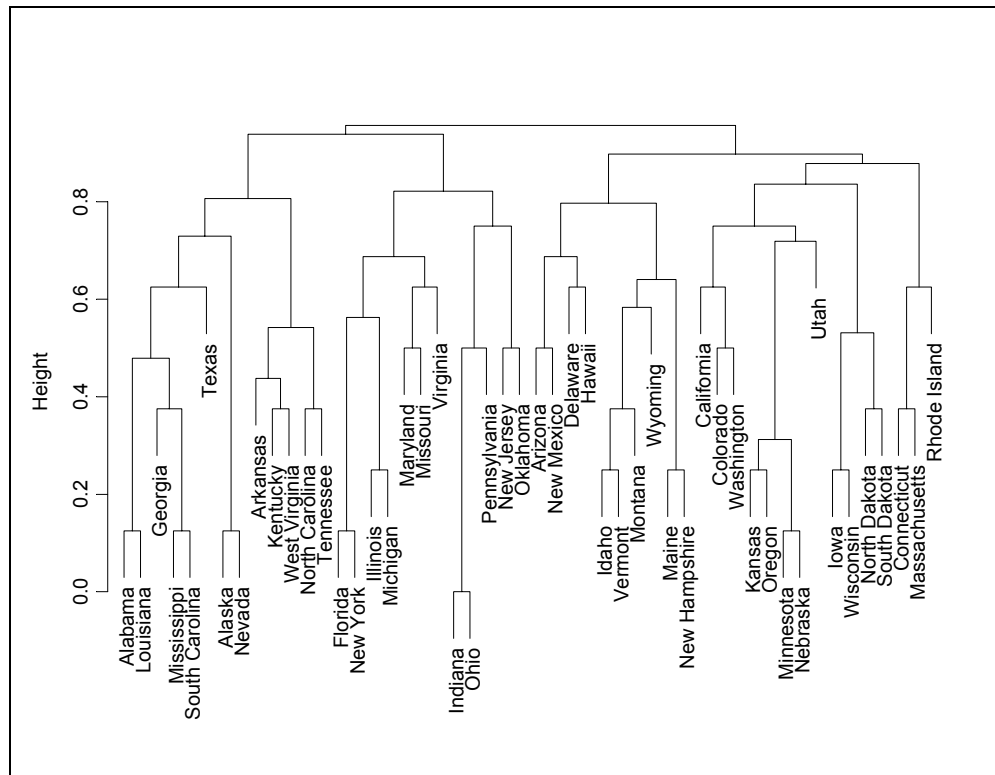


Figure 2 A Hierarchical Clustering Dendrogram of U.S. States from the S-PLUS [14] AGNES Clustering Algorithm

There are two ways to conduct hierarchical clustering: top down (divisive), and bottom up (agglomerative). A divisive clustering algorithm begins with $k=1$, with all n observations in one cluster. The clustering consists of splitting the data into smaller and smaller groups based on some similarity (or dissimilarity) measure, until $k=n$. An agglomerative clustering algorithm works the opposite way,

beginning with n clusters containing one observation each, then repeatedly combining similar clusters until $k=1$.

A clustering scheme with either $k=n$ or $k=1$ is not very useful in most cases, so the user must select the appropriate number of clusters by “pruning” the hierarchy to a meaningful size. This has traditionally been considered a separate problem, distinct from clustering itself. As will be shown later, Clementine’s Two Step hierarchical clustering algorithm automatically selects the appropriate number of clusters.

3. Self-Organizing Maps

A self-organizing map, or SOM, is described in [10] as “a constrained version of K-Means clustering.” This method is closely related to principal curves and surfaces, and has the similar benefit of reducing high-dimensional data to one- or two-dimensional space for data visualization.

Teuvo Kohonen, a Finnish mathematician, developed a popular algorithm to construct a SOM, fittingly called a Kohonen map. A Kohonen SOM builds on a two-dimensional $q_1 \times q_2$ grid lying in the principal component plane of the data. There are $K = q_1 q_2$ intersections in the grid, each one containing a “prototype” or representative observation, analogous to the initial cluster centers of a K-Means model. Each intersection has a two-dimensional “address” (Q_1, Q_2) , where $Q_1 \in \{1, 2, \dots, q_1\}$ and $Q_2 \in \{1, 2, \dots, q_2\}$. Each prototype m_j has an associated label $l_j \in Q_1 \times Q_2$, where $j \in \{1, 2, \dots, K\}$.

Taking for example n observations with p interval-scaled variables, the basic Kohonen algorithm processes each observation x_i one at a time, and finds the closest (Euclidean distance in $\mathbb{R}^p$) prototype m_j . For each prototype m_k within the neighborhood of m_j , m_k is moved toward x_i by this update:

$$m_k \leftarrow m_k + \alpha(x_i - m_k),$$

where α is a learning rate coefficient which decreases either linearly or exponentially at each step through the data. A prototype m_k lies in the neighborhood of m_j if $|l_j - l_k| < r$, where r is a distance threshold which decreases on each iteration.

This process is repeated iteratively until predetermined stopping criteria are met, with α and r decreasing on each iteration. The result is displayed as a two-dimensional grid of prototypes and their associated observations, which can be interpreted as a mapping or folding of the original p -dimensional data space onto $\mathbb{R}^2$. Figure 3 illustrates how prototypes that are closer together tend to contain more similar observations. Furthermore, the “folding” of the data space means that each corner is also “close” to its opposite.

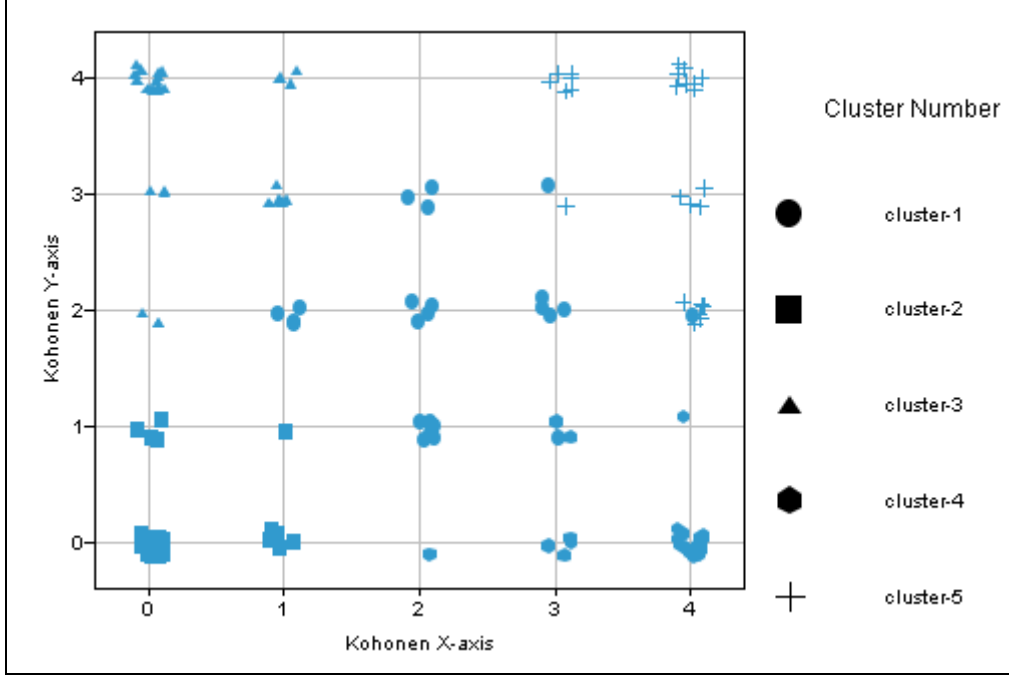


Figure 3 5x5 Kohonen Map of Generated Cluster Data

D. EVALUATION OF CLUSTERING RESULTS

1. Optimum Number of Clusters for K-Means

When applying a partitioning clustering algorithm such as K-Means, the number of desired clusters must be selected before clustering. Because cluster analysis is typically used to describe natural groupings in a data set, it is valuable to be able to calculate the optimum or “true” number of clusters, denoted by k^* . Hastie *et al.* propose in [10] a method to approximate k^* by within-cluster dissimilarity W_k as a function of k . W_k is a measure of within-cluster dissimilarity such as total sum of squares, total variance, or Root-Mean Squared Standard Deviation.[15] As k increases (the data is partitioned into more, smaller clusters), W_k will decrease (the clusters become more homogeneous). Once the optimum number

of clusters is exceeded, this increase in homogeneity (decrease in W_k) will be less pronounced, because clusters of similar objects are being divided into smaller groups of very similar objects. Thus, an approximation of k^* is the smallest value of k where this “kink” or flattening of the curve exists. Figure 4 shows this graph for the generated data in five distinct clusters, and $k^* = 5$ is quite readily apparent.

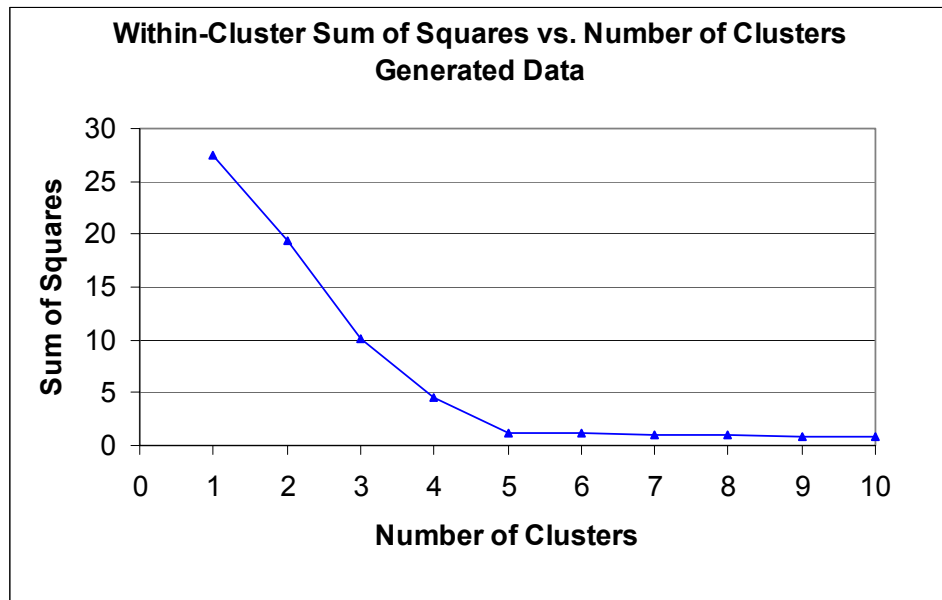


Figure 4 Evaluation of Optimum Number of Clusters

2. Cluster Validation

When building a supervised model, such as a classification tree, it is standard practice to validate the trained and tested model on a subset of the data that has not been previously “seen” by the model. There are different measures of performance for these models, such as misclassification rate, etc. In the unsupervised case, however, the true clustering arrangement of the data is

typically unknown, so there is no response variable with which to compare cluster membership for validation. How, then, does one evaluate clustering results? This question is particularly difficult in high-dimensional space, where visualization is not possible.

The most obvious (but least rigorous) solution to the cluster validation problem, and one which is appropriate in many contexts, is “does it work?” In other words, does the similarity among clustered objects make sense to an expert? While this is not a very objective measure of performance, it can be a good first step.

Gordon describes a more rigorous method in [8]. The data set of interest is randomly divided into two sets, called A and B . Set A is clustered using the model to be validated, and then the observations in B are “mapped” to the clusters found in A . Call this mapping B' . Set B is then clustered by the same method used to cluster A , with the same number of clusters. The final step is to compare the cluster membership of B with B' to determine the “co-clustering rate” of the model. In a “perfect” model, each observation would have the same cluster membership in both B and B' . This “co-clustering” is easily examined by forming a $k \times k$ contingency table for B and B' , as shown in Table 2. Assuming the cluster labels are arbitrary, it is usually possible to rearrange the columns of this table so that the “best” cluster mapping lies along the main diagonal, as shown in Table 3.

		B clusters				
B' clusters		cluster-1	cluster-2	cluster-3	cluster-4	cluster-5
	cluster-1	11	0	0	0	2
	cluster-2	0	17	0	0	0
	cluster-3	0	0	0	0	12
	cluster-4	0	0	10	0	0
	cluster-5	1	0	0	10	0

Table 2 A/B Validation Contingency Table for Generated Cluster Data

		B clusters				
B' clusters		cluster-1	cluster-2	cluster-5	cluster-3	cluster-4
	cluster-1	11	0	2	0	0
	cluster-2	0	17	0	0	0
	cluster-3	0	0	12	0	0
	cluster-4	0	0	0	10	0
	cluster-5	1	0	0	0	10

Table 3 A/B Validation Contingency Table for Generated Cluster Data (Rearranged to Illustrate Cluster Mapping on Main Diagonal)

In the case of perfect co-clustering, all of the off-diagonal entries would be zero. However, in any real clustering problem, the co-clustering will not be perfect. How then to analyze the “goodness” or validity of the chosen clustering model?

Conover in [5] discusses various techniques to measure dependence and association between the rows and columns of an $r \times c$ contingency table. The co-clustering problem described above lends itself well to this, and Cramer’s Coefficient has been chosen as the measure of association for our analysis.

	j = 1	j = 2	...	...	j = c	
i = 1	O ₁₁	O ₁₂	...	...	O _{1c}	r ₁
i = 2	O ₂₁	O ₂₂	...	...	...	r ₂
...	...	...	...	...	...	...
...	...	...	...	...	...	r _{r-1}
i = r	O _{r1}	O _{r2}	...	...	O _{rc}	r _r
	c ₁	c ₂	...	c _{c-1}	c _c	N

Table 4 Example of $r \times c$ Contingency Table

Cramer's Coefficient is developed as follows, referring to Table 4: given a contingency table with r rows and c columns, with row sums $r_1, r_2, \dots, r_r$ and column sums $c_1, c_2, \dots, c_c$, the observed value of cell (i, j) is denoted O_{ij} , and its estimated expected value (assuming independence of the rows and columns) is defined as $E_{ij} = \frac{r_i c_j}{N}$. The chi-square test statistic commonly used for testing the null hypothesis of independence in contingency tables, is

$$\text{defined as } T = \sum_{i=1}^r \sum_{j=1}^c \frac{(O_{ij} - E_{ij})^2}{E_{ij}}.$$

Cramer's Coefficient is the square root of the ratio of the observed value of T to the maximum possible value of T for a contingency table with the same number of observations and rows/columns, or $CC = \sqrt{\frac{T}{N(q-1)}}$. N is the number of observations, and q is the minimum of r and c . For our purposes, either r or c will do, as $r=c=k$ when comparing clustering results for A/B validation. For the generated clustering data validation shown in Table 2, $CC =$

0.945. This can be interpreted as a "94.5% clustering model," which would be assumed to be "better" than, say, a 75% clustering model.

Cramer's Coefficient has two properties that make it desirable as a comparative measure: first, it is dimensionless and unit-scaled ($0.0 \leq CC \leq 1.0$); and second, it is scale-invariant in O_{ij} , r_i and c_j .

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CLEMENTINE DATA MINING SOFTWARE

A. OVERVIEW

Clementine data mining software, produced by SPSS, Inc., is a robust tool that enables the user to quickly and easily determine relationships within large data sets through supervised and unsupervised modeling. IR Seaside uses Version 7.0, which is more user-friendly and intuitive than previous editions.

Clementine is used to analyze data by building what is called a *data stream*, or simply a *stream*, a sequence of operations that begins with a data source, flows through one or more *nodes* where the data is manipulated by field or record operations, and ultimately is used to build some sort of model. Output can be to a file, plot, or table.

Clementine uses three main data types, Sets, Flags, and Ranges, which are analogous to those discussed in Chapter III, Section A. Set fields are analogous to nominal variables (numeric or non-numeric); Flag fields are binary variables (either 1-0 or some other coding scheme) and Range fields are interval scaled variables.

B. UNSUPERVISED LEARNING MODEL TYPES

1. K-Means

The K-Means Node in Clementine produces a partition of the input data into k clusters. This type of model is intended for interval-scaled (Range type) data, but it will also accept categorical (Set and Flag type) variables by use of data transformations, discussed below. Figure 5 shows an example of a K-Means modeling dialog box, where

the user specifies k as well as other parameters for model building. Input fields may be selected at build time, or the model can use the Type Node settings, which are found upstream.

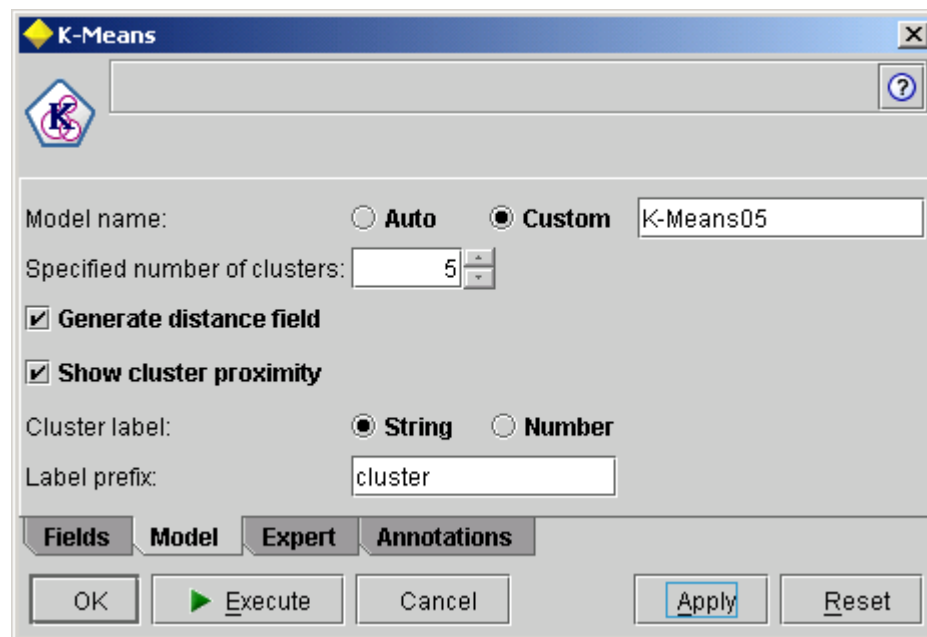


Figure 5 K-Means Node Model Options Dialog Box Model Tab

The "Expert" options available for building a K-Means model control the stopping criteria for the iterative cluster refinement process (number of iterations or change criteria) and encoding values for Set fields. The default encoding value of 0.70711 is approximately equal to $\sqrt{0.5}$, which properly weights the recoded Flag fields to produce a distance of 1.0 between observations with different values. Values closer to 1.0 weigh Sets more heavily than numeric fields. Figure 6 shows an example of the Expert Tab.

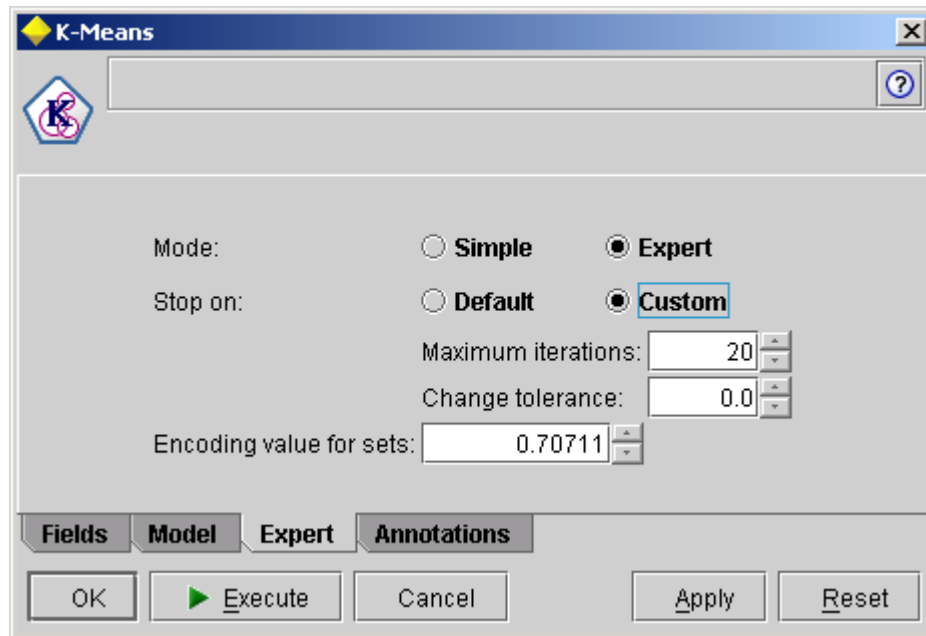


Figure 6 K-Means Node Model Options Dialog Box Expert Tab (Default Values)

K-Means executes a “quick cluster” algorithm that clusters numeric data very quickly and efficiently. The algorithm makes three passes through the data. In the first pass, initial cluster centers are selected. The second pass updates the initial cluster centers, and the final pass reassigns cases to the nearest cluster. Euclidean distance is used to determine “closeness.”

Binary (Flag type) variables are coded as 0 and 1, and their values are treated as continuous by the algorithm. This leads to some shortcomings, which are enumerated in the following section.

K-Means handles nominal (Set type) variables by recoding them into 1-0 Flag variables and treating them as described in the preceding paragraph. The transformation is undertaken by creating one dummy Flag field for each

possible value of the original Set field. For example, suppose a Set field "PMT_TYPE" has possible values of A, B, and C. K-Means creates three dummy Flag fields, PMT_TYPE_A, PMT_TYPE_B, and PMT_TYPE_C. Thus a record with PMT_TYPE of A would have PMT_TYPE_A = 1, PMT_TYPE_B = 0, and PMT_TYPE_C = 0.

When a K-Means Node is executed, the result is a "nugget" that represents the model. This model can be browsed to examine the number of clusters, number of records placed into each cluster, inter-cluster proximities, input fields, model build settings, and model training summary information. When data is clustered by the generated model, two new fields are created for each record: \$KM-<model name>, the cluster assignment label, and \$KMD-<model name>, the distance from each record to its cluster center. For example, a model named KMeans01 will have resulting fields \$KM-KMeans01 and \$KMD-KMeans01. Figure 7 is an example of browsing a K-Means model nugget, showing the cluster results of this particular model.

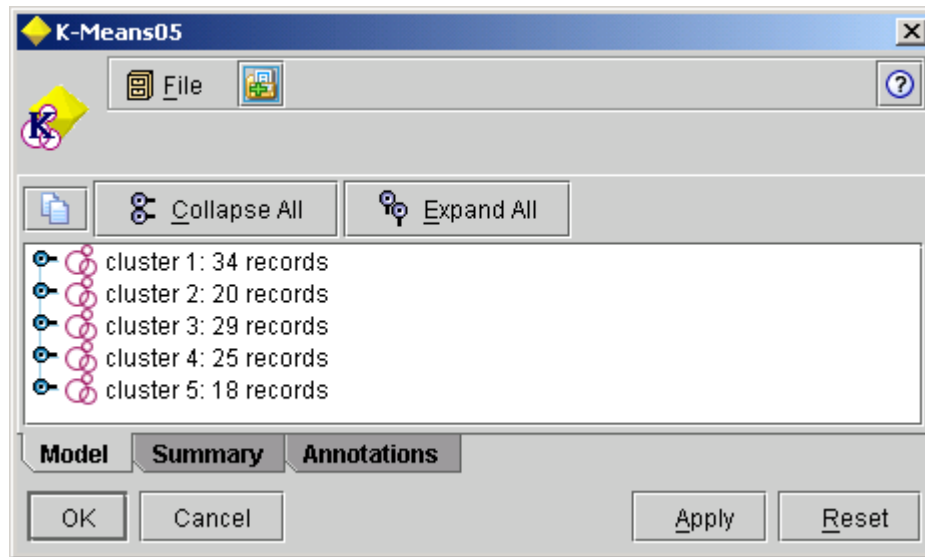


Figure 7 K-Means Generated Models Dialog Box Model Tab

2. Two Step

The Two Step Node in Clementine produces a hierarchical clustering of the data set. The user can either specify the number of clusters or allow the Two Step algorithm to automatically determine the appropriate number. There are no Expert options per se, but there are options to standardize numeric fields and exclude outliers. Figure 8 shows an example of the Two Step model building dialog box.

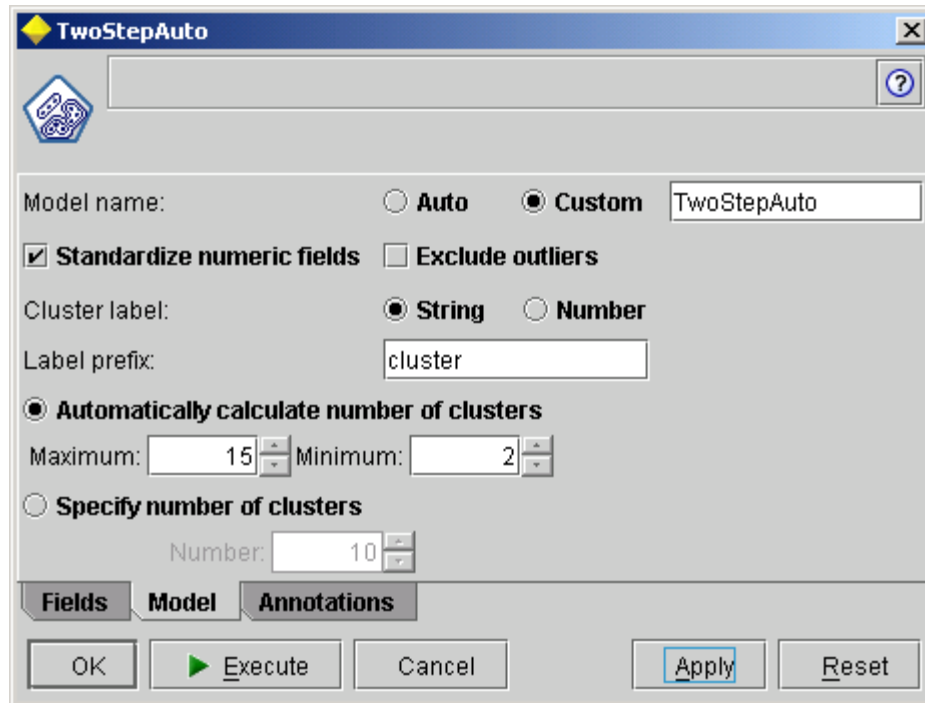


Figure 8 Two Step Node Model Options Dialog Box Model Tab

Two Step uses a log-likelihood function as a distance measure, and agglomeratively produces hierarchical clusters out of “dense regions” of records. The two steps implied by the name of the algorithm are pre-clustering and cluster membership assignment. Each step entails one pass through the data. The pre-clustering step consists of sequential examination of the data records, determination of dense regions, and tabulation of cluster features. After completion of the first data pass, the appropriate number of clusters (if not user-selected) is determined by finding the minimum Bayesian Information Criterion (BIC) value measured at each merge in the pre-clustering step, and refining it based on the ratio change in distance between the two merging clusters. The clustering step comprises a second pass through the data, during which each data record

is assigned membership in the cluster that is closest in terms of the log-likelihood distance measure. A more detailed discussion of the Two Step algorithm is available in [3].

Executing a Two Step node results in a browsable nugget. Browsing the model, as shown in Figure 9, reveals the same type of information as described in the preceding paragraph for a K-Means model. When clustering data with the generated Two Step model, a cluster label is assigned to each record in the form \$T-<model name>. Because the distance measure for Two Step is based on a likelihood function, no distance field is generated.

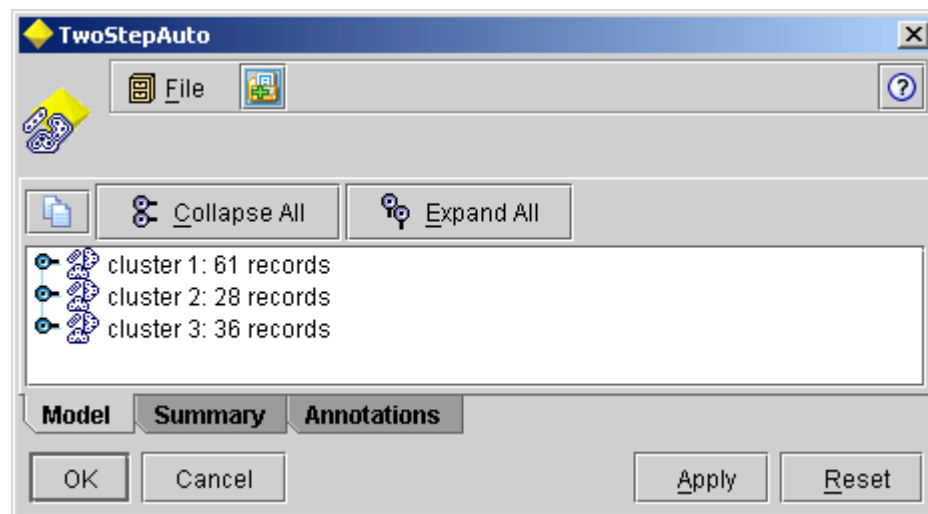


Figure 9 Two Step Generated Models Dialog Box Model Tab

3. Kohonen

The Kohonen node in Clementine essentially uses Kohonen's algorithm as described in Chapter III to produce a two-dimensional mapping of the data set. The only difference is that Clementine's algorithm does the mapping

in two phases. The first phase comprises rough estimation to capture gross patterns in the data; the second phase refines the mapping to finer detail. Figure 10 shows the “Simple” model options available, which essentially control stopping criteria and reproducibility.

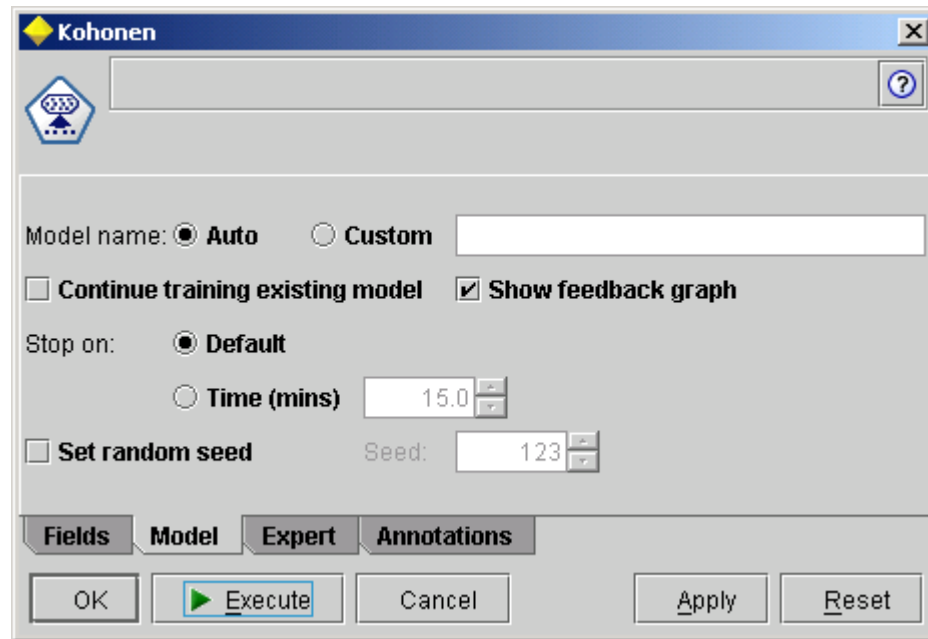


Figure 10 Kohonen Node Model Options Dialog Box Model Tab

The Expert options, shown in Figure 11, give the user much more control over the details of the Kohonen mapping. They allow selection of the map’s dimensions and the learning parameters discussed in Chapter III. The “neighborhood” parameter corresponds to the radius parameter r ; “eta” represents the learning rate parameter α ; and the “number of cycles” is the stopping criteria for the iterative process.

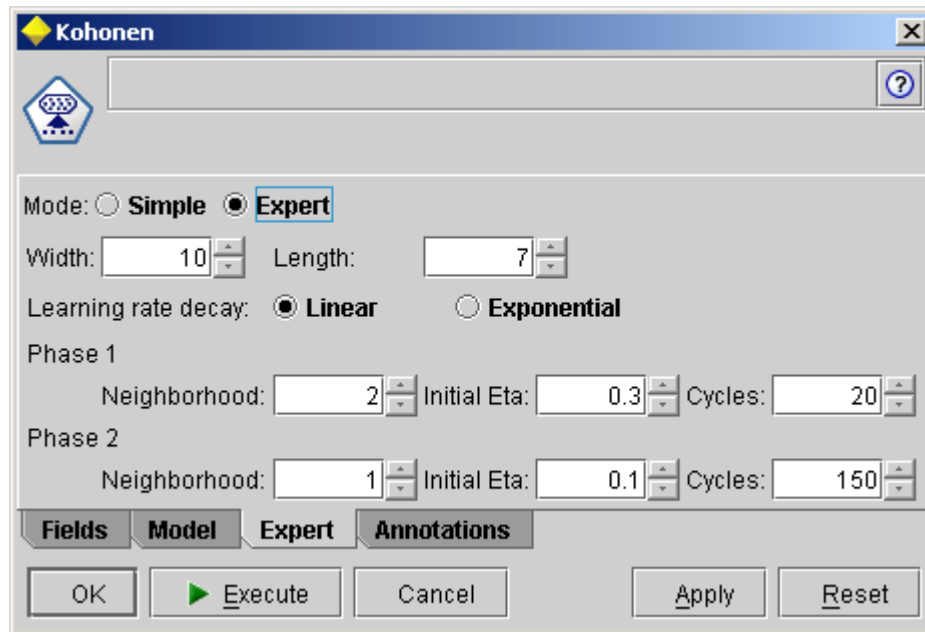


Figure 11 Kohonen Node Model Options Dialog Box Expert Tab

Unlike the K-Means and Two Step Generated Model Dialog Box, browsing the Kohonen nugget does not reveal much useful information other than the dimensions of the mapping. Because a Kohonen model is a type of neural network, there is a strong element of the “black box” to its function. This kinship with neural nets is also evident in the number of input and output neurons shown under “Analysis” on the Summary Tab shown in Figure 12.

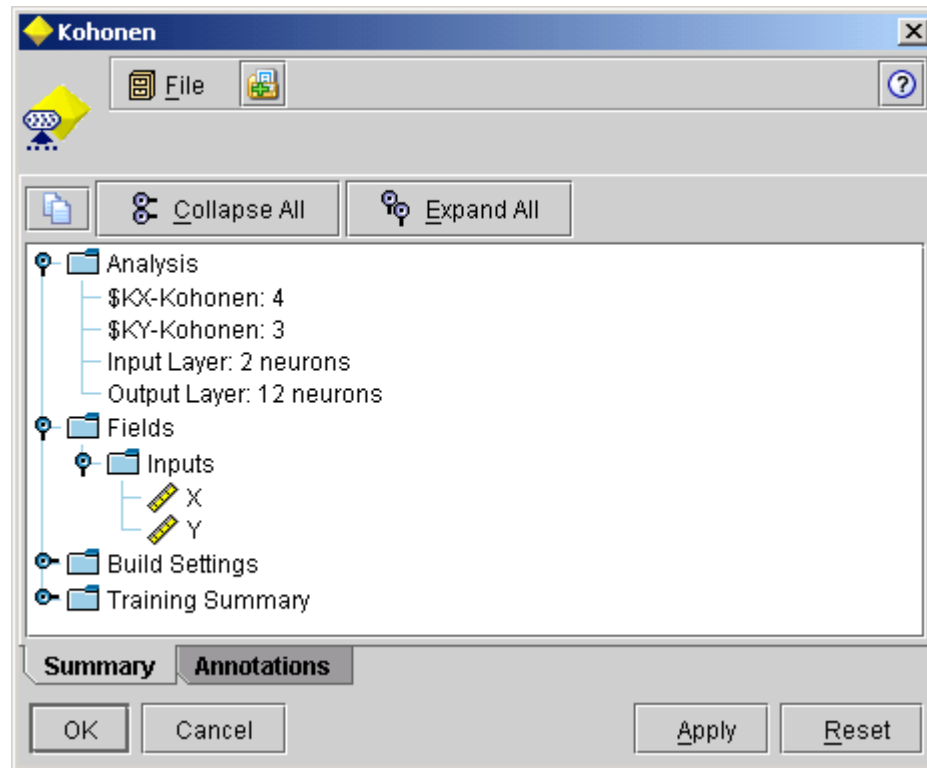


Figure 12 Kohonen Generated Models Dialog Box Summary Tab

As with K-Means and Two Step, a Kohonen model generates label fields for each record of a data set that is passed through it. In this case the fields are \$KX-
<model name> and \$KY-
<model name>, representing the (X,Y) coordinates of the prototype or node to which each record “belongs.”

C. SHORTCOMINGS OF CLEMENTINE UNSUPERVISED MODELING

Clementine’s data mining process is for the most part very user friendly and intuitive. However, there are several caveats and issues that must be understood to successfully conduct unsupervised modeling. Most of these required the assistance of SPSS Technical Support to thoroughly resolve.

The following issues, if heeded and understood, will greatly expedite the unsupervised modeling process and lower the modeler's average frustration level.

1. Although K-Means provides a mechanism for clustering categorical data, SPSS experts recommend against it. Their reason is that the clustering results obtained by using the K-Means algorithm on binary data tend to be arbitrary and are strongly dependent on the order of data presentation. See Appendix B for an example of this phenomenon.
2. The Two Step algorithm requires complete data for model building. If there are missing values in the data used to build a model, those records will be ignored. Missing values in data which are clustered by an existing Two Step model may result in cluster label assignment of \$null\$.
3. Kohonen modeling can be very memory- and time-intensive, particularly with large data sets. Changing any of the learning parameters may aggravate this problem to the point where the time required to build a large Kohonen map is excessive.
4. K-Means models sometimes cannot be browsed unless all data fields are converted to String and then re-typed. This is accomplished in the Basic Filter & Type Supernode discussed in Chapter V.

THIS PAGE INTENTIONALLY LEFT BLANK

V. FINDING CLUSTERS IN N02 VENDOR PAYMENT DATA USING CLEMENTINE UNSUPERVISED MODELS

A. OVERVIEW OF THE PROCESS

The unsupervised modeling procedure is divided into three parts: first, data pre-processing to select the data used for modeling; second, developing cluster models for the data set using each of the three types of Clementine unsupervised modeling nodes; finally, combining and comparing clustering output to select candidate records for examination. The next six subsections describe this process from database access through analysis. Detailed screen shots and other documentation can be found in Appendices A and B.

B. SOURCE DATA: THE N02 POPULATION DATABASE

The source data for development of the improved methodology is the Microsoft Access database table called N02_STA_POP2000/Population. The data is introduced into each stream using an SQL node to select the pre-established Open Database Connectivity (ODBC) connection. Appendix A shows the modified Operation Mongoose "Fields to Use" matrix for the N02 data, which lists each field, its description, and the status of the field (not used, model input, or analysis only).

The population data being clustered is organized by contracts. Each unique combination of the fields PIIN (Procurement Identification Number) and DEL_ORD (Delivery Order) comprises a unique contract. The "null hypothesis" for clustering these transactions is that all the

transactions from a particular contract will fall into the same cluster, and that transactions (and contracts) falling into the same cluster are similar. The ultimate objective of this process is to identify the "orphan" transactions (ones that don't fall into the "home cluster" for their parent contract) for further inspection by an auditor. A similar notion applies to Kohonen models, and is discussed in section F of this chapter.

C. DATA PRE-PROCESSING: THE BASIC FILTER & TYPE SUPERNODE

The intent of applying unsupervised modeling to vendor pay is to use as much of the data as possible, but certain fields have incomplete and/or unusable data that will not contribute to successful clustering, and must be excluded. The Basic Filter & Type Supernode, shown in Figure 13, pre-processes the data in order to avoid data-related problems with model building, and generates two new fields to be used for analysis. Screen Shots showing the details of each node's configuration are found in Appendix B.

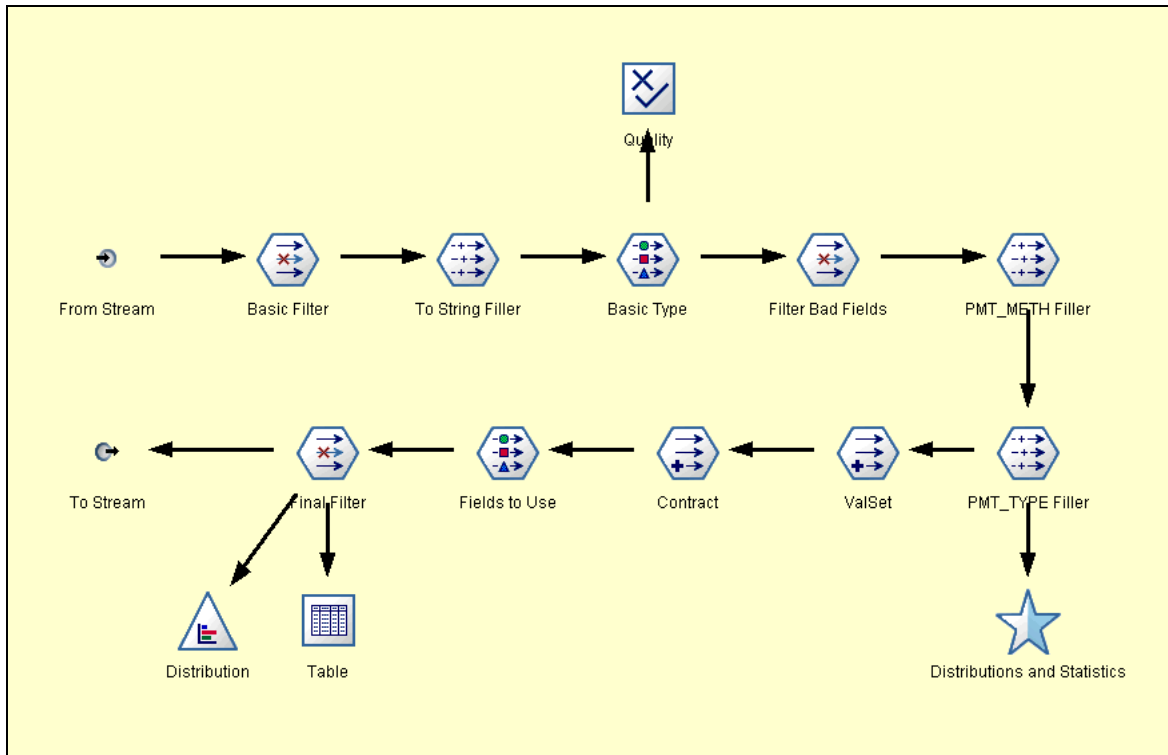


Figure 13 Basic Filter & Type Supernode

The Basic Filter Node removes the fields marked “N” in the modified “Fields to Use” matrix shown in Appendix A. In general these are fields that are never used for modeling. The To String Filler node converts each field to a String, then the Basic Type Node re-types each field to the appropriate Type. This step is necessary to ensure the browsability of generated models, as mentioned in Chapter IV, Section C.

The remaining fields are examined using the Quality Node and the Distributions and Statistics Supernode. These two steps identify “problem” fields, denoted by an “B” in Appendix A, that are filtered from the stream by the Filter Bad Fields Node; for example, Flag or Set fields that have only one value and all fields with less than 50% valid

entries. The two fields PMT_METH and PMT_TYPE have the value \$null\$ for four records. Because they exceed the 50% quality threshold, they are dealt with by replacing \$null\$ with "Blank" in the PMT_METH and PMT_TYPE filler nodes.

The ValSet Derive Node creates a validation set label field based on the field RNDM_NUM, which is used to separate the data set into two random partitions for cluster validation. The Contract Derive Node creates a single field concatenating the fields PIIN and DEL_ORD for ease in finding and manipulating records belonging to a unique contract.

Several fields are either not always appropriate as model inputs but have utility for model comparison and analysis, or are only used as model inputs in certain cases. These fields, marked "A" in Appendix A, are not completely filtered from the data stream, but rather their "Direction" is set to "none" in the Final Type Node. The Final Filter Node removes the remaining unusable fields (marked "F" in Appendix A) identified by the Distributions and Statistics Node or preliminary modeling efforts.

The end result "out" of this Supernode is 63 fields set as modeling inputs, of all three field types Range, Set, and Flag. There are also 14 other fields available for modeling or analysis downstream whose default direction is "none" in the final Type Node.

D. K-MEANS MODELING: KMEANS_UNSUP_POP_GWR.STR

1. Methodology

As described in Chapter III, Sections C and D, a "good" K-Means cluster model can be selected by evaluating

cluster dissimilarity as a function of number of clusters to find the optimum value of k, and then validated using the A/B random partitioning and cluster correspondence technique. The stream Kmeans_unsup_pop_GWR, shown in Figure 14, is used to generate K-Means models and output necessary to evaluate them. Screen shots with details of each node are found in Appendix B.

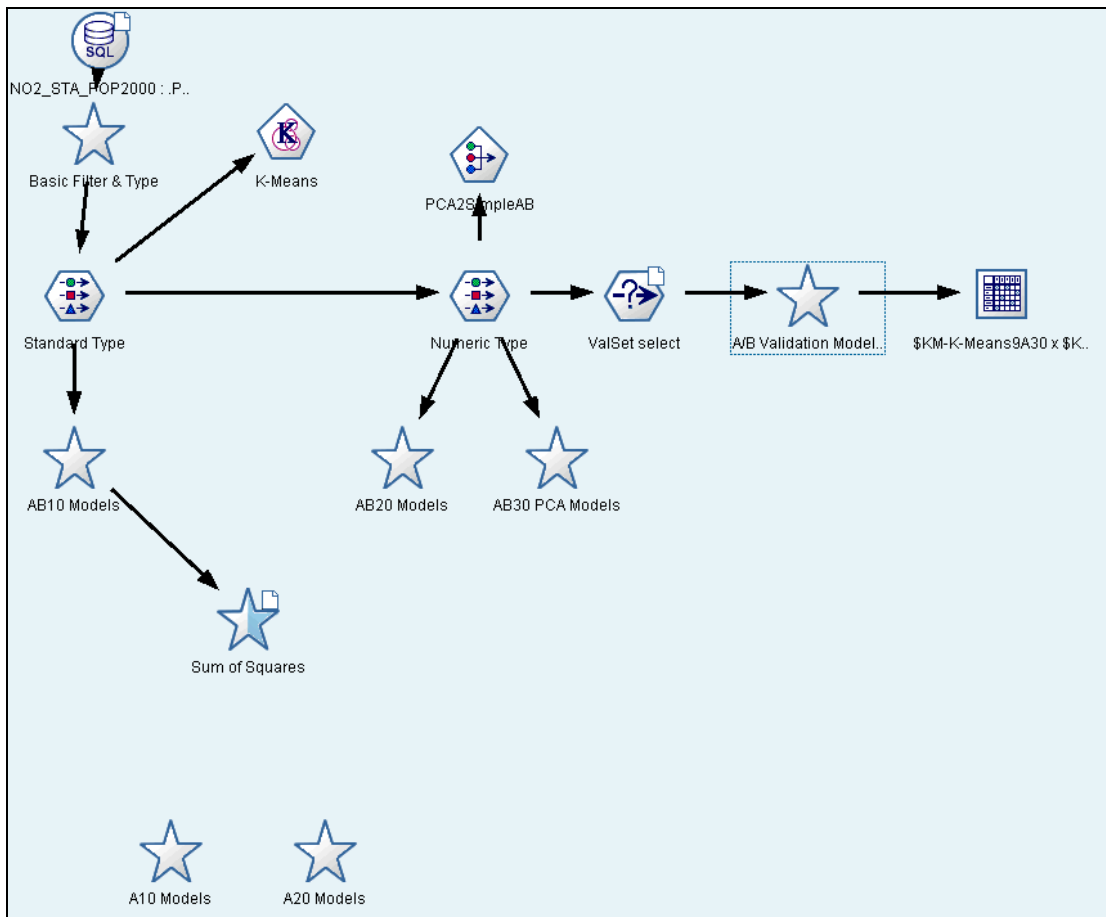
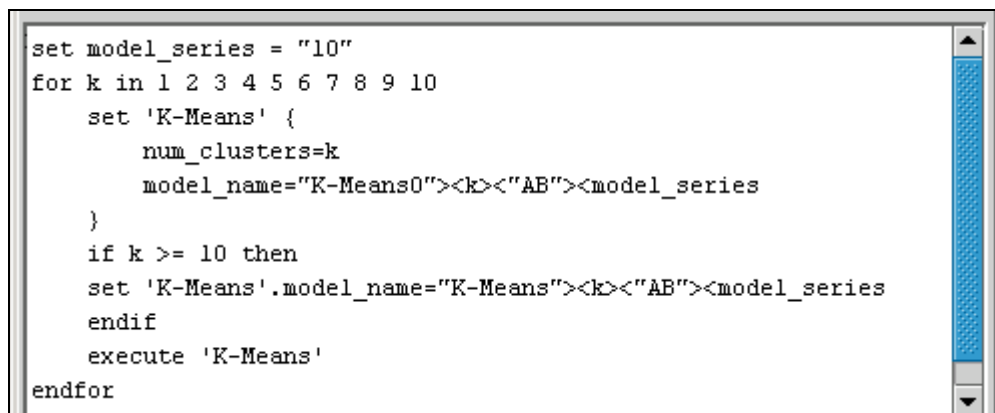


Figure 14 Kmeans_unsup_pop_GWR Main Palette

As described earlier in this chapter, the source data is brought in from the database, pre-processed in the Basic Filter & Type Supernode, then typed using the Standard Type Node. The Numeric Type Node selects only “Range” type fields for modeling the numeric-only models.

The first step in the model building and validation process is to build K-Means clustering models for all desired values of k . This is accomplished by a K-Means Node with Simple (default) settings, producing cluster membership label \$KM-<Model Name> and distance field \$KMD-<Model Name> for each record. The K-Means models are named in the format K-meanskkVVnn, where kk is the number of clusters (01, 02,...) ; VV is the validation set (A or B; AB denotes the entire data set); and nn is the model series (10 is all fields, 20 is numeric fields, 30 is numeric fields Principal Component Analysis). For example, K-Means05AB20 is a five-cluster model of the entire data set using only the numeric fields.

Construction of these models can be extremely tedious, as each one takes on the order of 2 to 5 minutes to build, and typically the modeler is interested in values of k from 1 to 10, 15, or even 20. This stream incorporates a Clementine script, shown in Figure 15, to automate the process. In each case the KMeans modeling node is used to generate the model.



```
set model_series = "10"
for k in 1 2 3 4 5 6 7 8 9 10
  set 'K-Means' {
    num_clusters=k
    model_name="K-Means0"><k><"AB"><model_series
  }
  if k >= 10 then
    set 'K-Means'.model_name="K-Means"><k><"AB"><model_series
  endif
  execute 'K-Means'
endfor
```

Figure 15 Model Building Script

After all the desired models are constructed, the “optimum” value of k must be determined to select the “best” model. The method described in Chapter III is used, which involves examining a plot of the total sum of squares of the distance field for each model vs. k . The first (lowest value of k) “kink” or flat spot in the curve indicates the “optimum” value of k . The total sum of squares is derived in the Sum of Squares Supernode, detailed in Appendix B. The spreadsheet tool “Sum of Squares,” described in Appendix C, is used to produce the Sum of Squares vs. Number of Clusters plot.

Once the “best” model in terms of cluster homogeneity is chosen, it is validated by randomly partitioning the data set into two equal parts, validation sets A and B. Each set is then clustered with $k = k^*$ (the “optimum” value of k determined previously). Validation set B is then passed through the two models (K-Means06A20 and K-Means06B20, for example) and the clustering results are compared in a two-way contingency table. Cramer’s Coefficient is then calculated, and the model’s validity can be evaluated and compared with that of other models. The spreadsheet tool “Cluster Correspondence Analysis Template,” described in Appendix C, is used to calculate Cramer’s Coefficient.

2. All Fields: AB10 Models

One of the objectives of this thesis is to utilize more of the data fields in population data sets than is possible in the supervised modeling process. The majority of these fields are categorical, so it is not intuitive that they would be useful for the K-Means algorithm, which

was developed for numeric data. However, Clementine's K-Means modeling node admits categorical variables (Sets and Flags) as inputs by way of transformation as described in Chapter IV. The methodology described in the preceding paragraph, when applied to the data stream as it leaves the Basic Filter & Type Node, results in the graph shown in Figure 16. Comparing this graph to the one shown in Figure 4, it is obvious there is no "kink" in this curve, so it is impossible to determine the optimum number of clusters using this method with these results.

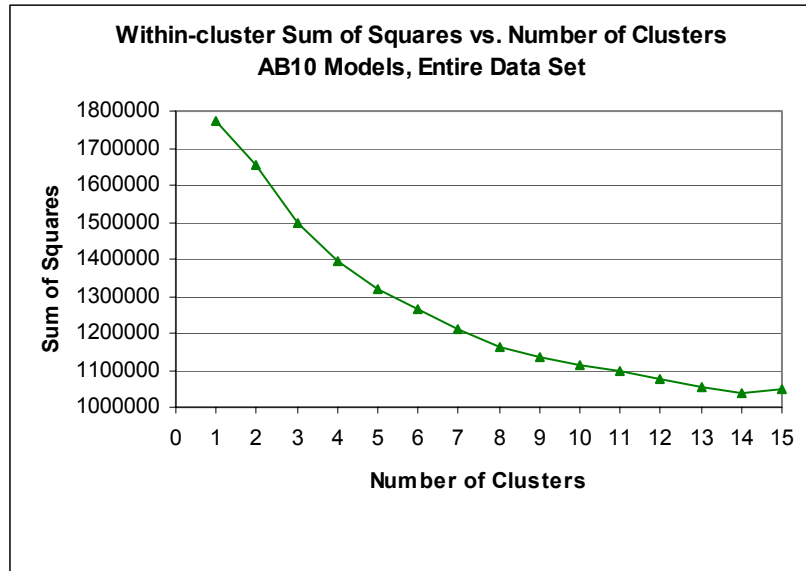


Figure 16 K-Means Models Built With All Fields

Furthermore, clustering models built with binary variables (Flag fields) tend to be arbitrary and are very sensitive to the ordering of the data. Reordering numeric data also changes the clustering results, but not as dramatically. Table 5 illustrates this difference; there is less similarity between the models built on the

categorical data than there is between those built on the numeric data. The details of this comparison appear in Appendix B.

		<i>Cramer's Coefficient</i>	
		Numeric, Reordered	Categorical, Reordered
		Kmeans06AB50	Kmeans06AB70
Numeric	Kmeans06AB20	83.03%	N/A
Categorical	Kmeans06AB60	N/A	72.12%

Table 5 Effect of Reordering Data on K-Means Models Built With Categorical Data

3. Numeric Fields Only: AB20 Models

The most logical approach to K-Means clustering is to use numeric variables (Range Type fields) only. The AB20 series models include all the Range fields included with the AB10 models, as well as several others that are used in place of the Set fields derived from them. Figure 17 indicates the proper number of clusters is six, and Cramer's Coefficient for this six-cluster model is 83.23%. Details of this model and calculation of Cramer's Coefficient appear in Appendix B.

This model, KMeans06AB20, is considered a "good" model and is included in the cluster analysis node as a tool to select interesting transaction records for further investigation.

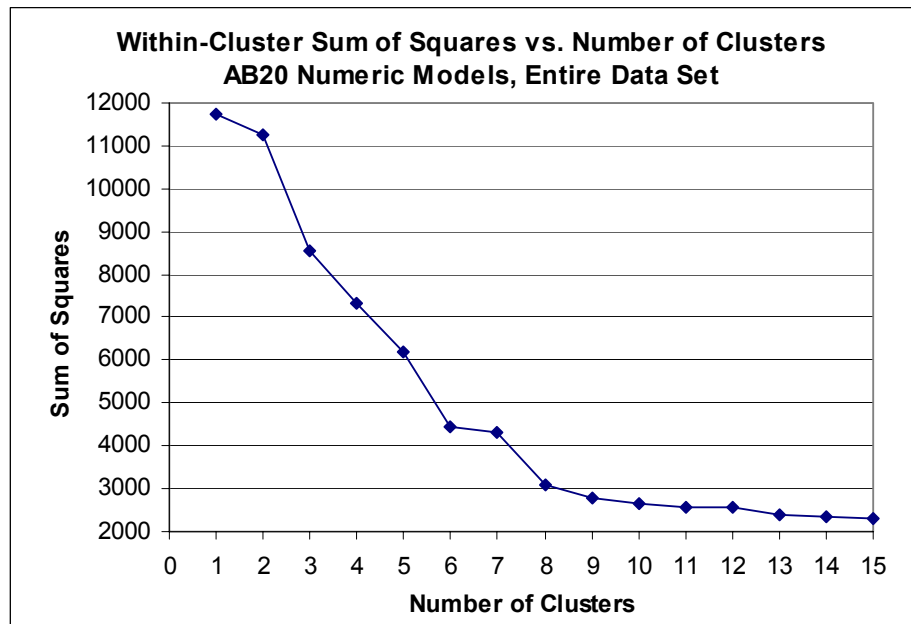


Figure 17 K-Means Numeric Only Models

4. Numeric Fields Only (Principal Components Analysis): AB30 Models

In an effort to further reduce the dimensionality of the dataset, Principal Components Analysis (PCA) was applied to the numeric fields, and the K-Means clustering process was then conducted on the resulting PCA-transformed data, resulting in the graph shown in Figure 18. Note that there are several “kinks,” but the first one is at $k=4$. This model, KMeans04AB30, has a Cramer’s Coefficient of 82.38%. Because it is not obvious that this is the correct number of clusters (there are also “kinks” at $k=6$ and $k=9$), and furthermore because the PCA transformation reduces the information available for clustering, this model was not selected as a tool for further analysis. Details of the PCA process and this cluster model appear in Appendix B.

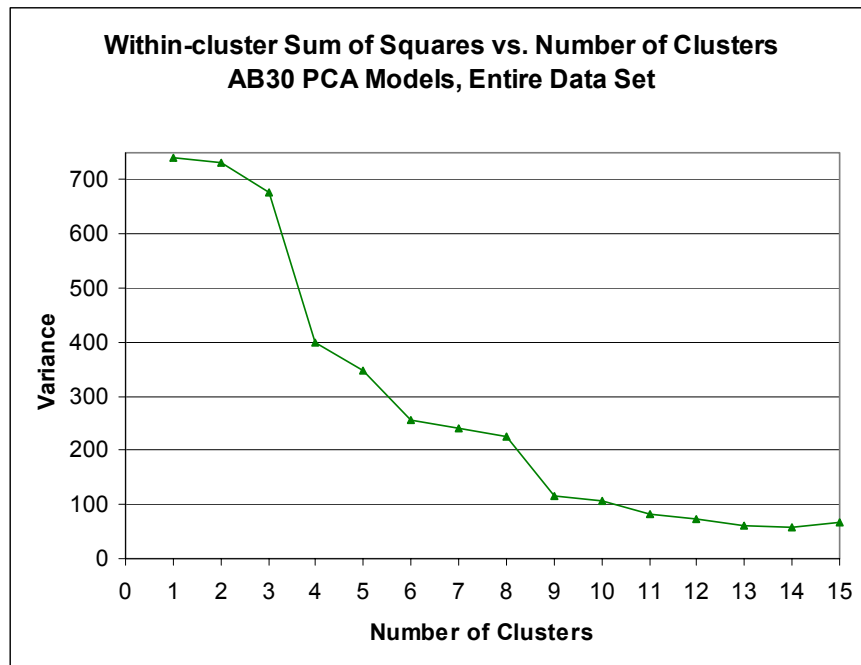


Figure 18 K-Means PCA Numeric Only Models

E. TWO STEP MODELING: TWOSTEP_UNSUP_POP_GWR.STR

Building a Two Step cluster model is not nearly as involved as for K-Means, because the “right” number of clusters is automatically selected by Clementine. The only real decision that needs to be made is the choice of input fields, which is already determined in the Basic Filter & Type Supernode. Figure 19 shows the stream palette. For Two Step models, the 20 series denotes the second iteration of modeling, rather than numeric only data.

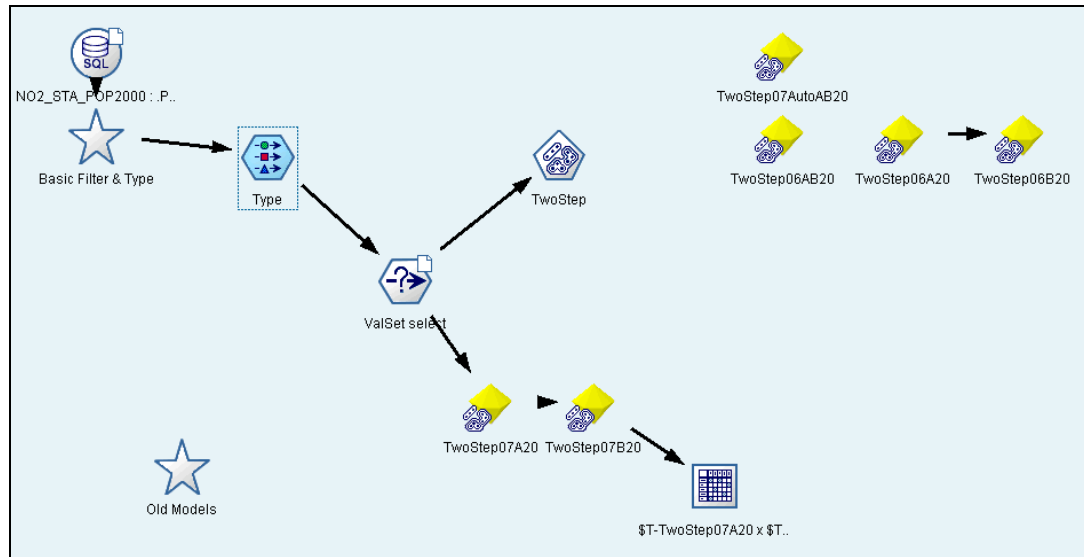


Figure 19 TwoStep_unsup_pop_GWR Main Palette

The model generated by the TwoStep node contained seven clusters, and is named TwoStep07AutoAB20. Its Cramer's Coefficient is 91.96%. For the sake of comparison, a six-cluster model, TwoStep06AB20, was also constructed; its Cramer's Coefficient is 90.83%. This indicates that when all the fields are included, the appropriate number of clusters is seven. The seven-cluster Two Step model is included as a tool for cluster analysis and selection of interesting records. Details of this stream, the model, and calculation of Cramer's Coefficient appear in Appendix B.

F. KOHONEN MODELING: KOHONEN_UNSUP_POP_GWR.STR

Building a Kohonen Self-Organizing Map is very straightforward compared to the K-Means modeling process described earlier. Figure 20 shows the Kohonen modeling stream, and details of the Kohonen node settings are given in Appendix B.

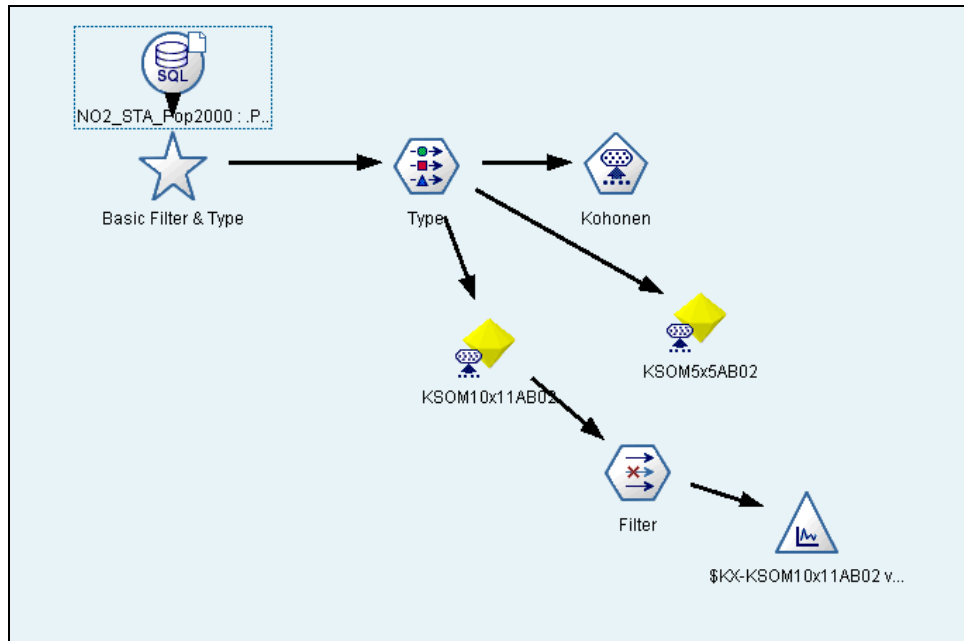


Figure 20 Kohonen_unsup_GWR Main Palette

Due to computer difficulties at Operation Mongoose, the only Kohonen models that were generated are a 5x5 map and a 10x11 map, shown in Figure 21 and Figure 22, respectively.

Interpretation of Kohonen mapping results is straightforward, as suggested by Abbott in [7]. Considering the fields \$KX-KSOM10x11AB02 and \$KY-KSOM10x11AB02 as "X" and "Y", one selects the records that are mapped to the "sparse" prototypes, such as the line along $Y = 1$ for $X = 5, 6, 7, 8, 9$, by generating a Select node for those values of X and Y. These records can then be evaluated using the analysis stream discussed in the next paragraph.

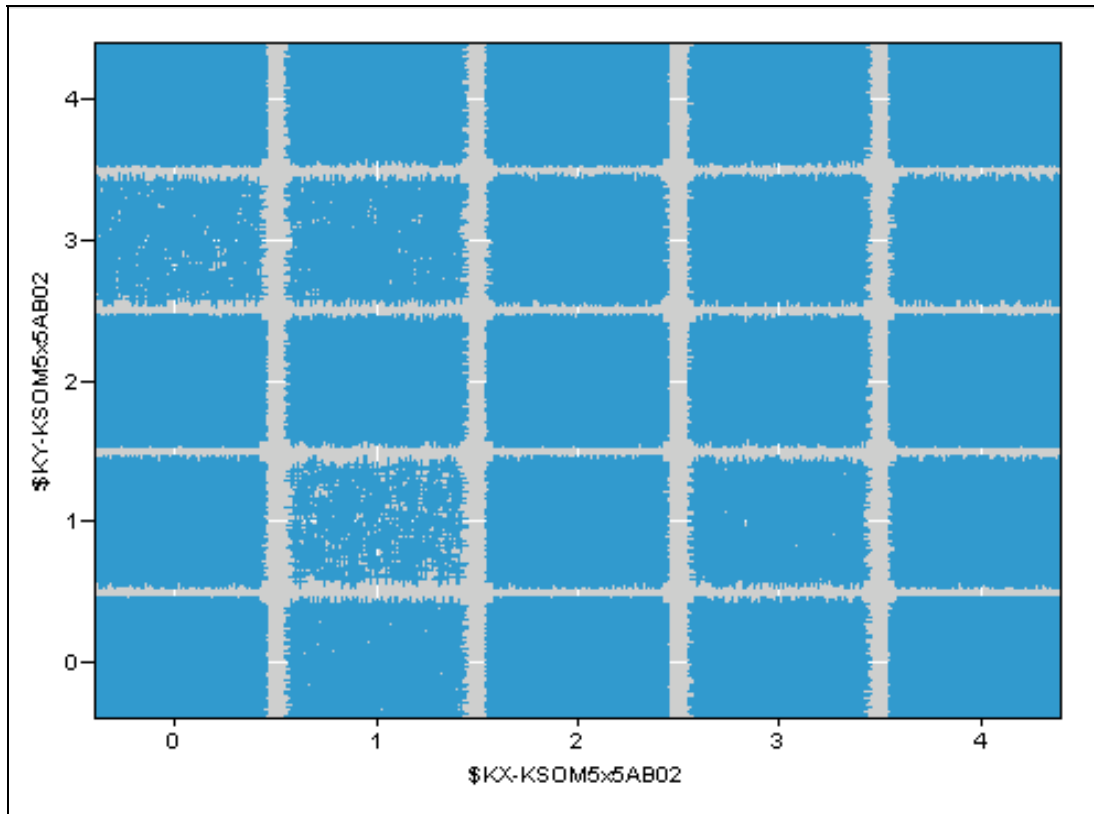


Figure 21 5x5 Kohonen Map

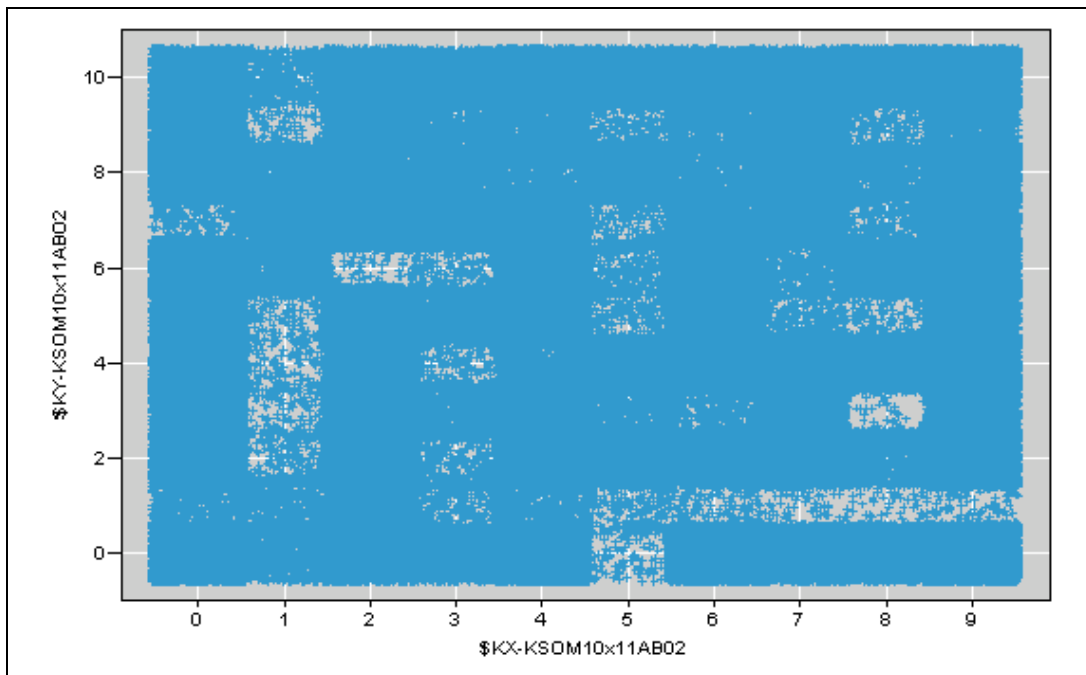


Figure 22 10x11 Kohonen Map

G. MODEL ANALYSIS: ANALYSIS_UNSUP_POP_GWR.STR

1. Overview

This stream uses the three generated models (KMeans06AB20, TwoStepAuto07AB20, and KSOM10x11AB02) to “vote” for the records to be further analyzed by a DFAS auditor. Figure 23 shows the main palette, whose output is a table listing the “interesting” transactions selected for further investigation by a DFAS auditor. The definition of an interesting transaction depends upon the type of clustering. The next few paragraphs detail the selection of interesting transactions for the different types of models.

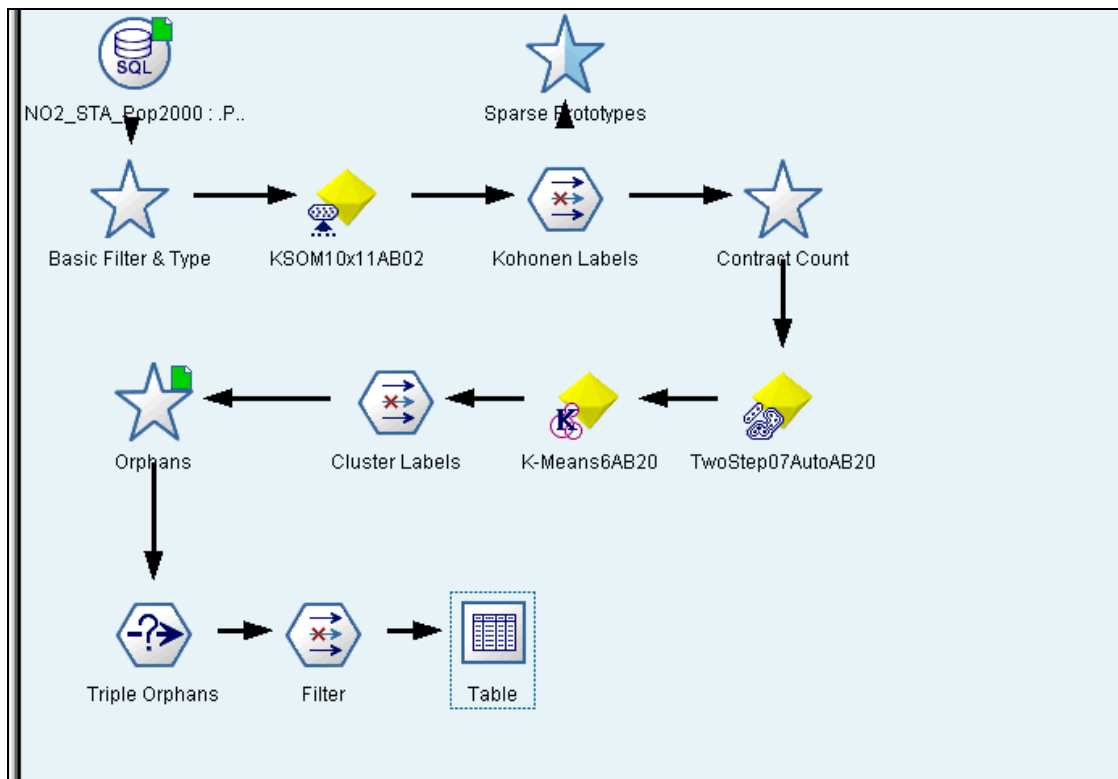


Figure 23 Analysis_unsup_pop_GWR Main Palette

2. K-Means and Two Step Models

For the K-Means and Two Step models, an interesting transaction is one that is an "orphan." An orphan is defined as a transaction that falls into a different cluster than the one containing 70% or more of the transactions belonging to its parent contract. The 70% level for this comparison was chosen to simplify the identification of home clusters and limit the number of orphans to be examined. For example, if 50% were the criteria for selecting home clusters, it is not obvious how to determine the home cluster of a contract whose transactions are evenly divided between two clusters.

3. Kohonen Maps

There are two ways to approach the task of identifying interesting transactions using a Kohonen map. The first is the one discussed in the previous section, selecting all the records assigned to the sparse prototypes of the map. Unfortunately the definition of sparse is completely subjective, so this selection can be arbitrary when done by "eyeballing" the Kohonen map. Other criteria are possible, of course, such as selecting all records assigned to prototypes less than a certain size, or a certain number of prototypes containing the fewest records. These are also arbitrary but less subjective than the "eyeball" method. Table 6 shows the results of this approach: KSOM_10 denotes the ten sparsest prototypes; KSOM_15 contains the fifteen sparsest ones; and KSOM<500 selects the 26 prototypes with fewer than 500 records assigned to them.

<u>Sparse Prototype Selections</u>	<u>Number of Transactions</u>
KSOM_10	1254
KSOM_15	2339
KSOM<500	6175

Table 6 Sparse Prototype Transaction Counts

The second possible approach uses the concept of orphans, allowing the Kohonen model to vote for transactions based on the same criterion as the K-Means and Two Step models. We identify the "home region" or group of prototypes for each contract, then proceed to find orphan transactions which do not fall into the same group of prototypes as the majority of the transactions for that particular contract. It is appropriate to identify a home region rather than a home prototype for a Kohonen map because with the large number of groups of transactions, it is likely that some of the contracts with thousands of records will be more or less equally divided along a line of prototypes. The mechanics of selecting orphan transactions is the same in this case as it was for the other two models: an orphan transaction is one assigned to a prototype containing less than a specified percentage of the transactions in the parent contract.

It is difficult to pick a percentage criterion for selection of orphan transactions that works well for all contracts. This was possible for the other two models because the maximum number of clusters occupied by transactions of any one contract was four. For the Kohonen

model, however, there are over 1,300 contracts whose transactions are assigned to four or more prototypes, and the maximum number of prototypes occupied by one contract is fourteen. The challenge is to select a percentage criterion that is small enough to ignore relatively large groups of transactions but still large enough to find orphans from contracts with a small number of transactions. As can be seen in Table 7, the choice of percentage criterion has an enormous impact on the number of records selected. This is an area in which further research is required, ideally to tailor the selection percentage criterion to the number of transactions in a contract as well as the number of prototypes among which those transactions are divided.

4. Implementation and Results

Each of the three clustering models selects its orphan transactions as described in the previous paragraphs, then the transactions selected by all three of the models are identified for further examination. Table 7 shows the number of orphan transactions for each of the three models, as well as the number of common records selected by various combinations of models. Note that three different percentage criteria for the Kohonen model, 30%, 20%, and 10%, are examined. The end result is that between 52 and 229 transactions are identified for physical examination by a DFAS auditor. Details of the orphan selection process and all the nodes shown in Figure 23 are given in Appendix B.

<u>Model Name</u>	<u>Number of Orphan Records</u>
KM	3494
TS	3665
Kohonen 30%	31,602
Kohonen 20%	16,612
Kohonen 10%	4299
KM & TS	229
Kohonen 30%, KM, & TS	155
Kohonen 20%, KM, & TS	97
Kohonen 10%, KM, & TS	52

Table 7 Orphan Transaction Distribution

THIS PAGE INTENTIONALLY LEFT BLANK

VI. TREE CLUSTERING

A. OVERVIEW

As proposed in [2] and described in Chapter I, Tree Clustering is a new unsupervised learning technique that exploits the properties of classification and regression trees to cluster data. This method is independent of variable type and includes automatic variable selection, automatic data scaling, and automatic selection of the optimum number of data clusters. This technique is implemented in S-PLUS by the function `tree.clust()`, detailed in Appendix B, which returns a dissimilarity matrix for further clustering by a conventional algorithm.

B. CLASSIFICATION TREES

1. Definition

Classification Trees are non-parametric supervised procedures to explain and/or predict a categorical response based on one or more input variables. The input variables can be categorical or numeric. Figure 24 shows an example of a classification tree that illustrates the following discussion. This tree is based on the S-PLUS Iris data, and it predicts the species (Setosa, Virginica, or Versicolor) of a flower based on its sepal length and width and petal length and width.

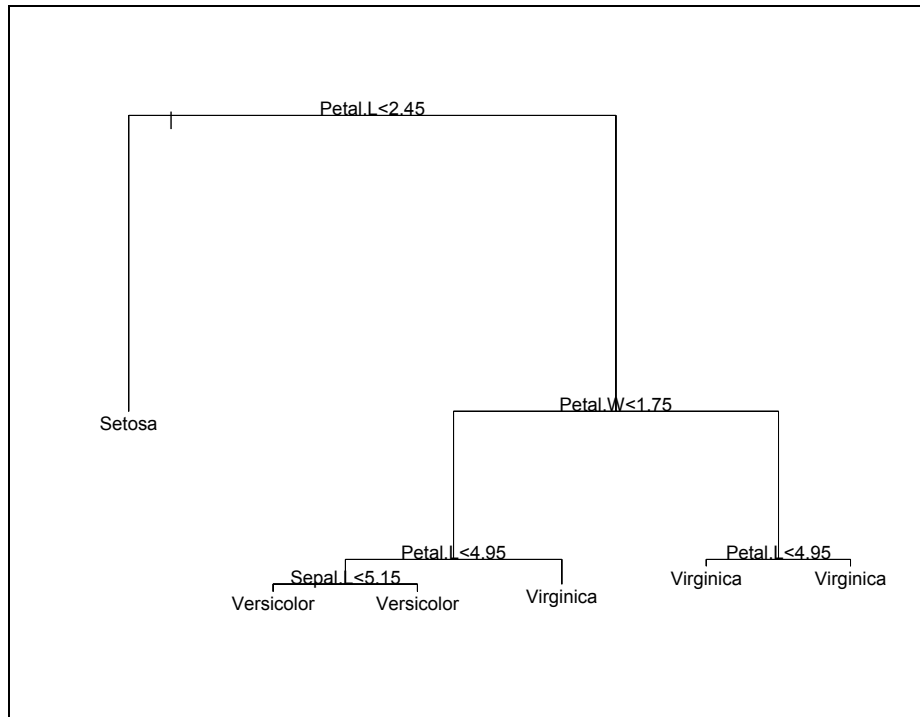


Figure 24 Classification Tree of the S-PLUS Iris Data

2. Construction

A classification tree is a binary splitting structure of the input data, beginning with a node containing all the data, called the root. The root is divided into two branches, each of which terminates in a node containing a subset of the data. These two nodes can each be divided into two branches, and so on. The terminal node of any branch is known as a leaf. Each of these node divisions is made by choice of a splitting variable and criteria to maximize the reduction in “node impurity” (in terms of predicted response) for the split. Theoretically (assuming each observation is unique), these divisions could continue until there are n leaves, one for each observation in the data set. In practice, however, excessive subdivision usually results in an overfit model. Therefore a tree is

typically “pruned” using cross-validation to obtain a model with reasonable predictive power that is not overfit. [1]

3. Node Impurity and Deviance

The concept of node impurity is important, because it is the basis for selection of splitting variables at each node. Deviance is a common measure of impurity; the higher the deviance in a node, the less related the observations are in terms of predicted response, and the higher the impurity. The following discussion from Holmes [11] is informative.

For a given classification tree of n observations having a response with K levels, the probability distribution of the response classes at leaf i is p_{ik} , $k=1,2,\dots,K$. For leaf i , the joint distribution of the number of observations of each of the K levels is multinomial with probabilities p_{ik} , $k=1,2,\dots,K$. The deviance at leaf i is defined as $D_i = -2 \sum_k n_{ik} \log(\hat{p}_{ik})$, where n_{ik} is the observed number of observations in level k , and $\hat{p}_{ik}$ is the maximum likelihood estimate of p_{ik} . The total deviance of the tree is equal to the sum of all the leaf deviances, $D = \sum_i D_i$.

As mentioned in the previous subsection, a classification tree is typically pruned to achieve a balance between predictive power and complexity. This pruning is usually accomplished by minimizing a complexity function such as $R_\alpha = R(T) + \alpha \cdot \text{size}(T)$, where $R(T)$ is a risk function that penalizes a high level of misclassification rate, impurity or some other measure of effectiveness,

$size(T)$ is a measure of complexity such as number of leaves, and α is a coefficient determining the weight given to the size function.

C. TREE CLUSTERING IMPLEMENTATION

In Tree Clustering, the similarity of a pair of observations is measured by the their tendency to fall in the same leaves of classification and regression trees. A classification tree is a type of supervised learning, and requires a response variable for its construction; however, in the clustering problem there is no such thing. Given n observations with p variables each, the Tree Clustering method sequentially constructs p trees, where the response variable of tree t is x_t for $t \in \{1, 2, \dots, p\}$. Each tree is "pruned" to its optimum size in terms of smallest cross-validated deviance. Each of the p trees can be described by its size (number of terminal leaves) and deviance (that is, decrease in overall deviance from the root to the terminal level). A tree with only one leaf (and thus zero deviance) suggests that its response variable contributes nothing to the similarity of observations. Likewise, these variables will likely not be chosen as "splitting" variables in other trees. Such a "noise" variable will be ignored entirely, which automatically limits selection variables to those with significant contribution to similarity.

After the trees are built, the distance between any two observations is proportional to the number of trees for which both observations fall in the same leaf. The label $L_t(i)$ denotes the leaf in tree t containing observation i . A

simple dissimilarity measure $d(i, j)$ between observations i and j is

$$d(i, j) = \sum_{t=1}^p d_t^{ij},$$

where d_t^{ij} is an indicator variable such that $d_t^{ij} = 1$ if $L_t(i) \neq L_t(j)$, and $d_t^{ij} = 0$ if $L_t(i) = L_t(j)$.

This dissimilarity measure is rather naïve, as it takes no account of the different degrees of dissimilarity among leaves of each tree. For example, two observations i and j in different leaves split from the same parent are presumably “less different” than i and k , which fall into leaves at different levels of the tree. Figure 25 illustrates this concept.

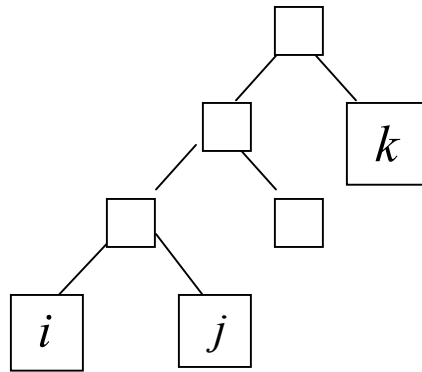


Figure 25 Classification Tree Illustrating Degrees of Dissimilarity

It is possible to overcome this shortcoming by implementing a more sophisticated dissimilarity measure. Using this measure, the distance between two observations in the same tree is the ratio of the change in deviance

obtained by trimming the tree back to their parent node (the lowest node containing both observations) to the overall deviance of the tree. For example, in the tree shown in Figure 25 the parent node of observations i and j is only one level up, so the change in deviance resulting from trimming the tree back to that node is very small. Therefore the deviance ratio is small, indicating that observations i and j are “close.” Considering observations i and k , however, the tree would have to be trimmed all the way back to the root, and the change in deviance would be equal to the deviance of the whole tree. This results in a deviance ratio of 1.0, indicating the two observations are very different. The overall dissimilarity between the two observations is the sum of their distances over all of the trees. This dissimilarity measure can be written as

$$d(i, j) = \sum_{t=1}^p \frac{\Delta_{s,t}^{ij}}{D_{s,t}},$$

where $\Delta_{s,t}^{ij}$ is the change in deviance resulting from trimming tree t back to the parent node s containing observations i and j , and $D_{s,t}$ is the deviance at the parent node s . The S-PLUS function `tree.clust()`, shown in Appendix C, calculates this distance for each (i, j) pair, then uses them to construct an S-PLUS object of type “dissimilarity” which can be used as an input to any of the S-PLUS clustering functions.

D. DEMONSTRATION OF THE TECHNIQUE

To demonstrate the variable selection capability of the Tree Clustering method, we add five “noise” variables

to the S-PLUS Iris data set (detailed in Appendix C). Additionally, to demonstrate the method's scale invariability, we multiply one of the original variables and one of the noise variables by ten, so they are one order of magnitude greater than the other variables. After running `treeclust()` on the noisy data, the resulting dissimilarity matrix is passed to `pam()`, a partitioning function in S-PLUS, with $k = 3$ clusters specified. Tree Clustering admits two of the noise variables and all of the original variables, and the PAM clustering results are as shown in Table 8, with Cramer's Coefficient = 92.22%.

<u>Species</u>	<u>Cluster 1</u>	<u>Cluster 2</u>	<u>Cluster 3</u>
Setosa	50	0	0
Versicolor	3	47	0
Virginica	2	3	45

Table 8 Contingency Table for Tree Clustering Scaled Iris Noise Data

To provide a standard for comparison, the scaled, noisy data was clustered using PAM with standardized variables. PAM standardizes variables (columns) by subtracting the column mean and dividing by the column standard deviation. These results are shown in Table 9, with Cramer's Coefficient = 70.58%. The tree clustering is clearly superior to the "straight" PAM clustering. Details of these results appear in Appendix C.

<u>Species</u>	<u>Cluster 1</u>	<u>Cluster 2</u>	<u>Cluster 3</u>
Setosa	48	2	0
Versicolor	0	17	33
Virginica	0	7	43

Table 9 Contingency Table for Clustering Scaled Iris Noise Data with PAM (Standardized Variables)

E. APPLICATION TO VENDOR PAYMENT DATA

To demonstrate the Tree Clustering method on a more complicated data set, we apply `tree.clust()` to the DFAS Knowledge Base. Use of the Knowledge Base provides a four-level response variable, `FRAUD_TYPE`, with which to evaluate the Tree Clustering results. There are 442 records, each with 43 input fields, used for this application.

Tree Clustering the Knowledge Base admitted all of the variables and resulted in the cluster assignments shown in Table 10, with Cramer's Coefficient = 27.33%. This clustering "score" is much lower than that obtained using the Iris data, probably because of a significant difference in the "true" classification of each data set. Specifically, while the classifications of the iris species is completely objective, the Knowledge Base transaction classifications are derived from (subjective) expert opinion of the type of fraud used to describe each transaction's parent case. Furthermore, the fraud experts originally proposed six classes of fraud, which were merged

into the four classes used today. The seemingly poor results perhaps reflect these two issues.

Fraud Class	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Big Systematic	144	107	29	2
Opportunistic	40	3	1	1
Piggyback	9	2	20	0
Small Systematic	46	17	20	1

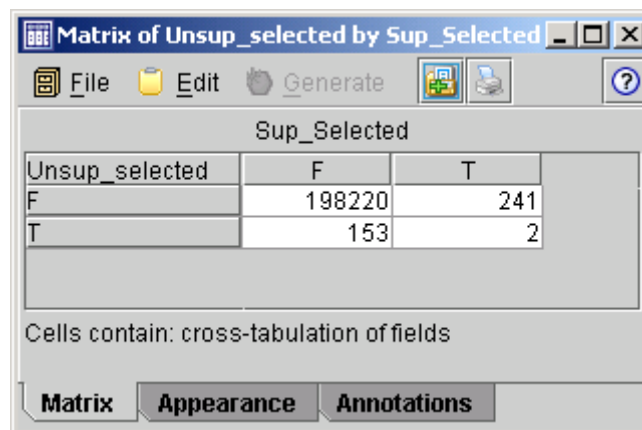
Table 10 Contingency Table for Tree Clustering Knowledge Base

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND RECOMMENDATIONS

A. SUPERVISED LEARNING VS. UNSUPERVISED LEARNING

One objective of using unsupervised modeling is to identify interesting transactions in the population payment data that might not be selected by the supervised modeling process. This can be evaluated by using a Derive Node to generate a Flag Field called Sup_Selected, whose value is "T" for the records selected by the supervised modeling stream. Likewise, a Derive Node is used to generate a Flag Field called Unsup_Selected for the records that are "triple orphans" in the unsupervised model analysis stream. Figure 26 clearly shows that the unsupervised methodology is selecting different records from the supervised process, as there are only two records selected by both.



Sup_Selected		
Unsup_selected	F	T
F	198220	241
T	153	2

Cells contain: cross-tabulation of fields

Matrix Appearance Annotations

Figure 26 Comparison of Records Selected by Supervised and Unsupervised Models

B. RELATIVE COMPARISON OF K-MEANS, TWO STEP, AND KOHONEN CLEMENTINE MODELS

Each of the three generated models, KMeans06AB20, TwoStepAuto7AB20, and KSOM10x11AB02 were used to identify

orphan transactions in the payment data. There are advantages and disadvantages to each type of modeling, summarized in Table 11.

<u>Model Type</u>	<u>Advantages</u>	<u>Disadvantages</u>
K-Means	• Method exists to select number of clusters • Handles missing values fairly well • Models can be built fairly quickly	• Limited to numeric data • Construction of models for Sum of Squares analysis is very tedious if not automated
Two Step	• Accepts categorical data • Only one model is required	• Missing values not allowed • Sometimes assigns \$null\$ as cluster label
Kohonen	• Easy to interpret in terms of sparse nodes	• Determination of orphan transactions can be difficult • Expert Settings require expert knowledge • Model building can be very time- and memory-intensive

Table 11 Comparison of Unsupervised Model Types

Using all three models to “vote” for candidate records for audit should capitalize on the strengths of each type of model while compensating for their weaknesses. Another approach could be to limit the voting to the K-Means and Two Step models, because their structures are very similar, although the clustering results are different. The Kohonen

results can then be used to identify records assigned to the sparse prototypes, and either treat them separately or include them in the voting scheme.

C. RECOMMENDATIONS FOR INTERNAL REVIEW SEASIDE

Appendix E contains the Recommended Standard Operating Procedure (SOP) for Unsupervised Modeling, which is intended to supplement the existing Datamining SOP. This SOP should be used together with the spreadsheet tools described in Appendix C to conduct unsupervised modeling with more rigor and success than is possible under the current process. Applying this methodology should enhance IR Seaside's ability to successfully identify records for audit that contain Conditions Needing Improvement or fraudulent payments.

Supervised Modeling should continue to be a part of IR Seaside's datamining toolbox. The single largest impediment to improvement of this process and successful detection of fraud is the age and incompleteness of the Knowledge Base. If at all possible, more current fraud cases should be obtained and used to update and expand the Knowledge Base. If this is not achieved, the supervised modeling process and results will not improve.

Finally, it is recommended to investigate the utility of modeling using the CNI database rather than the Knowledge Base as a model-training tool. This is an area ripe for further graduate research that could maintain the strong relationship between IR Seaside and the Naval Postgraduate School Operations Research Department.

D. TREE CLUSTERING WITH LARGE DATA SETS

Although the results shown in Chapter VI, Section D are not “good” in the sense of clustering based on fraud classification, they do demonstrate the utility of the Tree Clustering methodology on a relatively large data set. Using this method on a very large data set (for example, the NO2 data used to develop the unsupervised modeling methodology), will be limited primarily by the ability of the S-PLUS clustering algorithms to handle very large dissimilarity matrices. While the scalability of the `tree.clust()` function is theoretically unlimited, its performance is heavily dependent upon the size of the data set to be clustered, particularly in the number of observations.

There are two primary performance factors in Tree Clustering a very large data set: the number of trees constructed (a function of the number of variables, p), and the size of the dissimilarity matrix produced (a function of the number of observations, n). Using “Big O” notation, and assuming $n^2 \gg p$, the function `tree.clust()` runs in $O(n^2)$ time, because the dissimilarity between each pair of observations must be calculated. In the unusual case where $p > n^2$, the function will run in $O(np)$ time while building p trees with the n variables. The storage required is also $O(n^2)$, because the dissimilarity matrix (actually implemented as a vector), must contain entries for each pair of observations.

APPENDIX A. NO2 POPULATION DATABASE

g:\home\abott\dw\Clemlib\ProcessDocumentation\FieldsToUseNO2.xls Sup&UnsupTypeNodeFieldInputs rt&dc28may02				
Field Name	Transformation Description	NO2 Unsup	CLEM Type Node Type	Comment
I = Not filtered in Basic Filter & Type Supermode. These fields may be used as direct input for modeling or as a potential source for a Clementine derive node. A = Not filtered in Basic Filter & Type Supermode. These fields are used for analysis or record identification or as a potential source for a Clementine derive node, but not as a modeling input. B = "Bad field" filtered in the Basic Filter & Type Supermode. These fields are never used directly or indirectly for modeling input and are not used for analysis or record identification in the modeling process. N = Filtered in the Basic Filter & Type Supermode. These fields are never used directly or indirectly for modeling input and are not used for analysis or record identification in the modeling process.				
1 SUB_DT	Submission Date	N	Set	
2 SYS_ID	System ID	N	Set	
3 SITE_ID	Submission Site ID	N	Set	
144 FILE_SEQ	File Sequence Number	N	Typeless	
5 PIIN	Procurement Identification Number	A	Typeless	
6 DEL_ORD	Delivery Order	A	Typeless	
145 SYS_DCN	System Document Control Number	N	Typeless	
8 DOV_NUM	Disbursing Office Voucher Number	N	Typeless	
19 DOV_AMT	Disbursing Office Voucher Amount	N	Real Rg	
9 PMT_NUM	Payment Number	N	Typeless	
50 VOU_STAT	Voucher Status	B	Set	Unsupervised modeler may wish to use this set versus the vou_stat flags below.
13 CHK_DT	Check Date	I	Typeless	
15 DSSN	Disbursing Station Symbol Number	N	Typeless	
10 CHK_NUM	Check Number	N	Typeless	
12 CHK_AMT	Check Amount	I	Real Rg	
53 EFT_ACCT	EFT Account Number	N	Typeless	
54 EFT_RTN	EFT Routing Number	N	Typeless	
47 TIN	Tax identification number	N	Typeless	
55 DUNS_NUM	Data Universal Numbering System	N	Typeless	NAME CHANGED FROM 'DUNS'. Not populated in the KB
CAGE_CD	Contractor and Government Entity	N	Typeless	NEW FIELD Not populated in the KB
30 MAN_IND	Manual Indicator	N	Set	Use m_pymt flag
14 CHK_STAT	Check Status	N	Typeless	Are payment cdf records created when check status equals V?
CHK_CAN_DT	Date Check Cancelled	N	Typeless	NAME CHANGED FROM 'DTCKCAN' Not populated in the KB
57 CHK_XREF	Cross Reference Check Number	N	Typeless	
INT_PD_AMT	Interest Paid Amount	N	Real Rg	NAME CHANGED FROM 'IP_AMT' Not populated in the KB
46 TAX_AMT	Tax Amount	N	Real Rg	Not populated in the KB
16 DISC_AMT	Discount Amount	N	Real Rg	Use discount flag transformation.
18 LOST_AMT	Lost Discount Amount	N	Real Rg	Not populated in the KB. Could be used to derive set range field.
17 LOST_CD	Lost Discount Code	N	Set	Not populated in the KB
34 PMT_METH	Payment Method	I	Set	Not fully populated in the KB. Unsupervised modeler may wish to use this set versus the pmt_meth flags below.
35 PMT_TYPE	Payment Type	I	Set	Not fully populated in the KB. Unsupervised modeler may wish to use this set versus the pmt_type flags below.
33 PMT_CAT	Payment Category	I	Typeless	Not fully populated in the KB
36 PMT_PROV	Payment Provision	I	Set	Not fully populated in the KB. Unsupervised modeler may wish to use this set versus the pmt_prov flags below.
PPA_XMPT	Prompt Payment Act Exempt	I	Set	NEW FIELD Not populated in the KB
29 INV_AMT	Invoice Amount	I	Real Rg	
25 INV_DT	Invoice Date	I	Typeless	
26 INV_NUM	Invoice Number	N	Typeless	
27 INV_RCVD	Invoice Receipt Date	I	Typeless	
INV_ENTR_DT	Invoice Entered Date	N	Typeless	Use to derive date differences, INV_DT minus INV_ENTR_DT or INV_RECVD minus INV_ENTR_DT. Not populated in the KB.
22 FRT_STAT	Freight Status	N	Set	Not populated in the KB
7 LINEITEM	Line Item	N	Typeless	Contract Line Item Number
FOB	Freight on Board	I	Set	Not populated in the KB
21 FRT_AMT	Freight Amount	N	Real Rg	Not populated in the KB
MDSE_ACC_DT	Merchandise Acceptance Date	N	Typeless	Use to derive date differences, INV_DT minus INV_ENTR_DT or INV_RECVD minus INV_ENTR_DT. Not populated in the KB.
MDSE_DEL_DT	Merchandise Delivery Date	I	Typeless	Use to derive date differences, INV_DT minus INV_ENTR_DT or INV_RECVD minus INV_ENTR_DT. Not populated in the KB.
RMT_CD	Remit Code	N	Typeless	
RMT_NAME	Remit Name	N	Typeless	

Table 12 Modified Fields To Use Matrix (on four pages)

39	RMT_L1	Remit to Address Line 1	N	Typeless	
40	RMT_L2	Remit to Address Line 2	N	Typeless	
41	RMT_L3	Remit to Address Line 3	N	Typeless	
42	RMT_L4	Remit to Address Line 4	N	Typeless	Not populated in the KB
43	RMT_CITY	Remit to City	N	Typeless	
44	RMT_ST	Remit to State	N	Typeless	
45	RMT_ZIP	Remit to Zip Code	N	Typeless	
	BOO_ID	Base Contracting Office ID	N	Typeless	Not populated in the KB
101	AWARD_DT	Award Date	N	Typeless	
	CON_AMT	Contract Amount	N	Real Rg	Not populated in the KB. Could be used to derive ranges in unsupervised modeling.
	GS_IND	Goods/Service Indicator	I	Set	Not populated in the KB
	NET_VND	Net Vendor Days	N	Typeless	Not populated in the KB
	CON_STAT	Contract Status	I	Set	Not populated in the KB
	CON_TYP	Contract Type	I	Set	Not populated in the KB
	VND_NAME	Vendor Name	N	Typeless	Not populated in the KB
	VND_ADR1	Vendor Address 1	N	Typeless	Not populated in the KB
	VND_ADR2	Vendor Address 2	N	Typeless	Not populated in the KB
	VND_ADR3	Vendor Address 3	N	Typeless	Not populated in the KB
	VND_CITY	Vendor City	N	Typeless	Not populated in the KB
	VND_ST	Vendor State	N	Typeless	Not populated in the KB
	VND_ZIP	Vendor Zip Code	N	Typeless	Not populated in the KB
	VND_TYP	Domestic or Foreign Vendor	I	Typeless	Not populated in the KB
	VND_ID	Vendor Identification Number	N	Typeless	Not populated in the KB
	VE1_CD	Voucher Examiner Code 1	B	Typeless	Not populated in the KB
	VE2_CD	Voucher Examiner Code 2	B	Typeless	Not populated in the KB
	VE3_CD	Voucher Examiner Code 3	B	Typeless	Not populated in the KB
	VE4_CD	Voucher Examiner Code 4	B	Typeless	Not populated in the KB
	VE5_CD	Voucher Examiner Code 5	B	Typeless	Not populated in the KB
	SYS_UNIQ	System unique data not recorded	N	Typeless	Never used as input, information only.
	CDF_RMT_NAME	CDF Remit Name	N	Typeless	
	CDF_RMT_L1	CDF Remit to Address Line 1	N	Typeless	
	CDF_RMT_L2	CDF Remit to Address Line 2	N	Typeless	
	CDF_RMT_L3	CDF Remit to Address Line 3	N	Typeless	
	CDF_RMT_L4	CDF Remit to Address Line 4	N	Typeless	
	CDF_RMT_CITY	CDF Remit City	N	Typeless	
	CDF_RMT_ST	CDF Remit State	N	Typeless	
	CDF_RMT_ZIP	CDF Remit Zip Code	N	Typeless	
	PAYMENT	Consolidates transaction into a single payment	N	Typeless	Field used to consolidate transactions into a single payment
74	TRANS_NUM	Number of transactions associated with a single payment	N	Real Rg	
75	PAYEE	Complete name of Payee	N	Typeless	Never used as input! Required for results calculations! Leave in as Typeless/None!
143	PAYEE13	First 13 Characters of Payee	N	Typeless	
	ADDRESS13	First 13 Characters of Address	N	Typeless	
	C_INV_NUM	Cleaned Invoice Number	N	Typeless	Modify clean invoice number to drop leading zeros.
88	AGGREG_PAYEE	Total Dollar Amount Paid to a Specific Payee	N	Real Rg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling.
89	AGGREG_ADR	Total Dollar Amount Paid to a Specific Address	N	Real Rg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling.
102	INV_AWARD_DT	Number of days between invoice date and contract award date	N	IntegerRg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling? Also, consider boosting in Real Range is used for input for C5 model.
103	INV_REC_V_AWARD_DT	Number of days between invoice received date and award date	N	IntegerRg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling? Also, consider boosting in Real Range is used for input for C5 model.
104	CHK_AWARD_DT	Number of days between check date and award date	N	IntegerRg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling? Also, consider boosting in Real Range is used for input for C5 model.
105	INV_REC_V_INV_DT	Number of days between invoice received date and invoice date	I	IntegerRg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling? Also, consider boosting in Real Range is used for input for C5 model.
106	CHK_INV_DT	Number of days between check date and invoice date	I	IntegerRg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling? Also, consider boosting in Real Range is used for input for C5 model.
107	CHK_INV_REC_V_DT	Number of days between the check date and invoice received date	I	IntegerRg	Derive range sets or flags for modeling. May be able to use Real Range for NN modeling? Also, consider boosting in Real Range is used for input for C5 model.
60	INTEREST	Was Interest Paid	I	Flag	Caution: No occurrence in KB.
61	MILPAY	Military Pay Appropriation	I	Flag	Caution: Limited occurrence in KB.
62	DBOF	DBOF Appropriation	I	Flag	

63	BRAC	BRAC Appropriation	I	Flag	Caution: No occurrence in KB.
64	OTHERX	X Year Appropriation other than BRAC, DBOF, UNUSUAL	I	Flag	
65	UNUSUAL	Appropriation = 5188, 5189, 6875, 3880, 3875 or 8164	B	Flag	Caution: Limited occurrence in KB.
66	ALLX	All X year appropriations	I	Flag	
67	Y1_PRIOR	1 year Expired Appropriation	I	Flag	
68	Y1_CUR	1 Year current appropriation	I	Flag	
69	Y2_PRIOR	2 Year Expired Appropriation	I	Flag	Caution: Limited occurrence in KB.
70	Y2_CUR_1ST	2 Year Current Appropriation Paid 1st Year	I	Flag	Caution: Limited occurrence in KB.
71	Y2_CUR_2ND	2 Year Current Appropriation Paid 2nd Year	I	Flag	
72	Y3_PLUS	3 or more year appropriation	I	Flag	
73	ALL_OTHER	None of the above appropriations starting with MILPAY	I	Flag	Caution: Limited occurrence in KB.
76	CNT_CDF	CBE	N	Flag	Should never be used as input.
141	PAY_ORDER	Some version of 'Pay to the Order' in the Remit to field	N	Flag	Change in business practice should have eliminated this flag.
138	ENHANCE_PAYEE	Flag when Payee found in Remit_L1 field	I	Flag	
77	ORDER_CDF	Replace 'Pay to the Order' with Remit_L1	N	Flag	Should never be used as input.
79	STE	Pynt made to suite address	I	Flag	
80	LOCKBOX	Payments to lockboxes	B	Flag	Caution: No occurrence in KB.
81	POBOX	Payments to PO box	I	Flag	
82	INV_PAYEE	Payee with different invoice number lengths	I	Flag	
83	INV_CNT	Contract with different invoice number lengths	I	Flag	
84	DOV_AMT_2K	DOV_AMT >= to 2000	I	Flag	
85	DOV_AMT_1K	DOV_AMT >= to 1000	I	Flag	
86	AVG_5K	Average payment amount to payee is >= 5K	I	Flag	
87	PAYEE_4_PYMT	4 or more payments to the same payee	I	Flag	
90	MULTI_PAYEE	Multiple payees to the same address	N	Flag	Caution: Modeler should consider using derive node to combine with eft_payee. Ref: ?nod
91	MULTI_ADR	Multiple address to the same payee	N	Flag	Caution: Modeler should consider using derive node to combine with eft_adr. Ref: ?nod
92	INV_SEQ	Invoices out of sequence to the same payee	I	Flag	
93	PMT_FREQ_HI	Regular payments over a period of time	I	Flag	
94	PMT_FREQ_LO	Payments occurring in a short period to time	I	Flag	
95	TINS	Tax identification number is present in record	I	Flag	Notes: 1) Not fully populated in KB. 2) When tins flag = "1", tin is null!!!
96	MULTI_TINS	Multiple TINS for a Payee	I	Flag	Not fully populated in KB.
97	MULTI_PAYTIN	Multiple Payees to the same TIN	I	Flag	Caution: No occurrence in KB.
148	MULTI_PAYEE_K	Multiple Payees to the same contract	I	Flag	
149	MULTI_ADDR_K	Multiple Addresses to the same contract	I	Flag	Modeler should consider using derive node to combine with multi_eft_k. Ref: ?nod
150	MULTI_TINS_K	Multiple TINS to the same contract	I	Flag	Caution: Limited occurrence in KB.
151	MULTI_EFT_K	Multiple EFT to the same contract	I	Flag	Caution: No occurrence in KB. Modeler should consider using derive node to combine with multi_addr_k. Ref: ?nod
98	DISCOUNT	Was discount paid	I	Flag	Caution: Limited occurrence in KB. In NO2, 10,515.
99	M_PYMT	Manual Payment	I	Flag	
100	FEW_PYMT	Flag companies that have <200 payments in a year	I	Flag	
108	MISC_OBLIG	Flag that looks for MORD or MOD in the PIIN	I	Flag	
109	EFT_PAYEE	Multiple payees to same EFT	N	Flag	Caution: No occurrence in KB. Modeler should consider using derive node to combine with multi_payee. Ref: ?nod
110	EFT_ADR	Multiple EFTs to a single Payee	N	Flag	Caution: No occurrence in KB. Modeler should consider using derive node to combine with multi_adr. Ref: ?nod
134	DUPPAY102	Duplicate Payment Indicator 102 - Logic: Same PIIN, Same SPIIN, Same Inv#, Same Inv# - 6888	N	Flag	Note Dup pays are sparsely populated in KB/Possible vendor fraud

	DP109	Duplicate Payment Indicator 109 - Logic: Same K, Same Inv_Amt, Same Award Dt, Inv_Amt > 0	I	Flag	
	DP111	Duplicate Payment Indicator 111 - Logic: Same K, Same Inv_Amt, Inv_Amt > 0	I	Flag	
	NOT_DFAR	PIIN Del Ord does not conform to the DFAR	I	Flag	
140	FRAUD_TYPE	Knowledge Base: Big Sys, Small Sys, Piggy, Opportunistic Payment Population: Assured Not Fraud (NF)	N	Set	Never used as input. Always out during model creation.
139	SEQ_ID	Record Sequence ID Number	N	IntegerRg	
	NUMADR_K	Number of addresses (ADR_L1+QTY) to an individual contract (PIIN+DO).	I	IntegerRg	
	NUMEFT_K	Number of EFT addresses (ACCT+RTN) to an individual contract (PIIN+DO).	I	IntegerRg	Not populated in KB. Modeler should consider derived field combined w NUMADR_K
	NUMADREE	Number of addresses (ADR_L1+QTY) to a whole payee.	I	IntegerRg	
	NUMEFTTEE	Number of EFT addresses (ACCT+RTN) to a whole payee.	I	IntegerRg	Not populated in KB. Modeler should consider derived field combined w NUMADREE
	NUM_EE_K	Number of whole payees to an individual contract (PIIN+DO).	I	IntegerRg	
	MODELAWDT	Number of days between the K Award Date and the Merchandise Delivery Date.	N	IntegerRg	Suggest future range set. Actual number <u>may</u> be appropriate in Neural Net models?? Caution: Award Dt not always reliable.
	MODELCKDT	Number of days between the Check Date and the Merchandise Delivery Date.	I	IntegerRg	Suggest future range set. Actual number <u>may</u> be appropriate in Neural Net models??
	MODELINDT	Number of days between the Invoice Date and the Merchandise Delivery Date.	I	IntegerRg	Suggest future range set. Actual number <u>may</u> be appropriate in Neural Net models??
	MODELJRDt	Number of days between the Invoice Received Date and the Merchandise Delivery Date.	I	IntegerRg	Suggest future range set. Actual number <u>may</u> be appropriate in Neural Net models??
	NUMAWEE	Number of contracts (PIIN+DO) with the same award date to the same whole payee.	N	IntegerRg	Suggest future range set. Actual number <u>may</u> be appropriate in Neural Net models?? Caution: Award Dt not always reliable.
	NUMCHEE	Number of checks to the same whole payee on the same day.	I	IntegerRg	Suggest future range set. Actual number <u>may</u> be appropriate in Neural Net models??
	CASE	Knowledge Base Case Name	-	Typeless	Never used as input! Required for results calculations! Leave in as Typeless/None!
	RNDMNUM	Random number	A	Typeless	Created in population only not in Splits
	M_INV_AWARD_RG	Ranges of days between the invoice and	I	Set	Caution: Award Dt not always reliable
	M_INV_RECV_AWARD_RG	Ranges of days between the invoice received and award dates	I	Set	Caution: Award Dt not always reliable
	M_CHK_AWARD_RG	Ranges of days between the check and	I	Set	Caution: Award Dt not always reliable
	M_INV_RECV_INV_RG	Ranges of days between the invoice received and invoice dates	I	Set	
	M_CHK_INV_RG	Ranges of days between the check and	I	Set	
	M_CHK_INV_RECV_RG	Ranges of days between the check and invoice received dates	I	Set	
	M_STE1_OR_BOX1	STE flag = 1 <u>or</u> PCBOX flag = 1 <u>or</u>	I	Flag	
	M_DBOF1_OR_NDFAR1	DBOF flag = 1 <u>OR</u> NOT_DFAR flag = 1	I	Flag	
	M_DBOF0_OR_NDFAR1	DBOF flag = 1 <u>and</u> NOT_DFAR flag = 0	I	Flag	
	M_DBOF1_AND_NDFAR1	DBOF flag = 1 <u>and</u> NOT_DFAR flag = 1	I	Flag	
	M_DBOF0_AND_NDFAR1	DBOF flag = 0 <u>and</u> NOT_DFAR flag = 1	I	Flag	
	M_MADRK1_OR_MFTK1		I	Flag	
	M_MADR1_OR_EADR1	MULTI_ADR = 1 <u>or</u> EFT_ADR = 1	I	Flag	
	M_AGG_ADR_RG	Ranges of AGGREG_ADR amounts	I	Set	
	M_AGG_PAYEE_RG	Ranges of AGGREG_PAYEE amounts	I	Set	
	M_DOV_AMT_RG	Ranges of DOV_AMT amounts	I	Set	

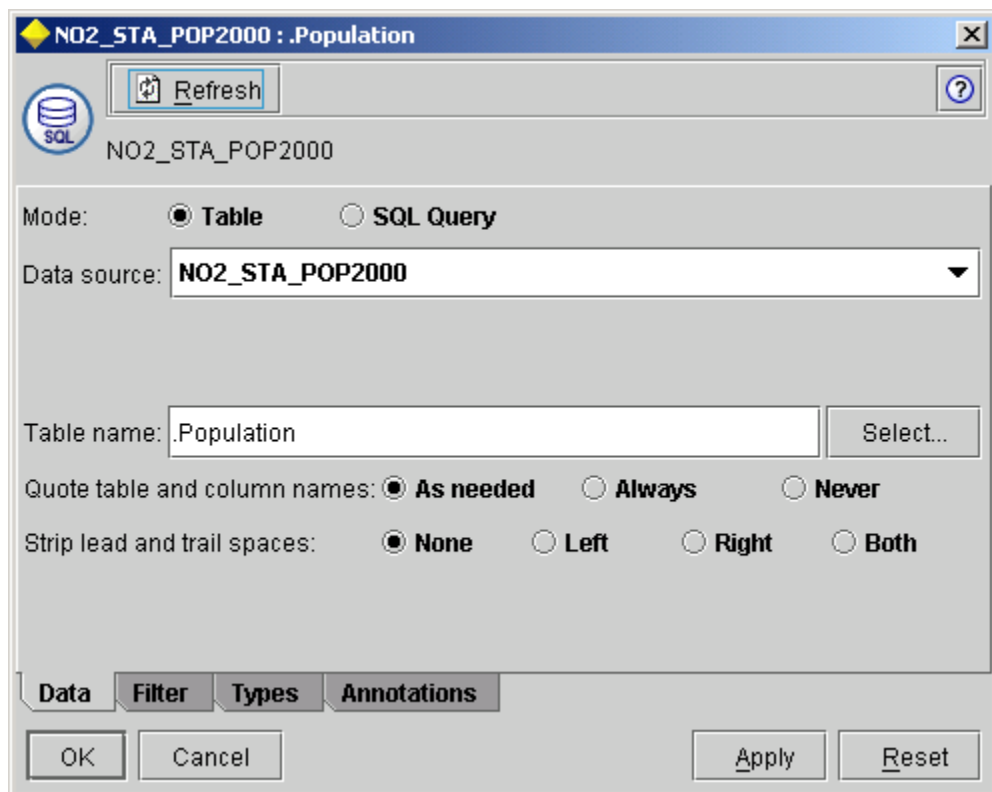


Figure 27 SQL Node Dialog Box for NO2_STA_POP2000 Database

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. NO2 POPULATION UNSUPERVISED MODELING IMPLEMENTATION AND DETAILED RESULTS

1. BASIC FILTER & TYPE SUPERNODE

As described in Chapter V, this Supernode pre-processes the data in preparation for clustering.

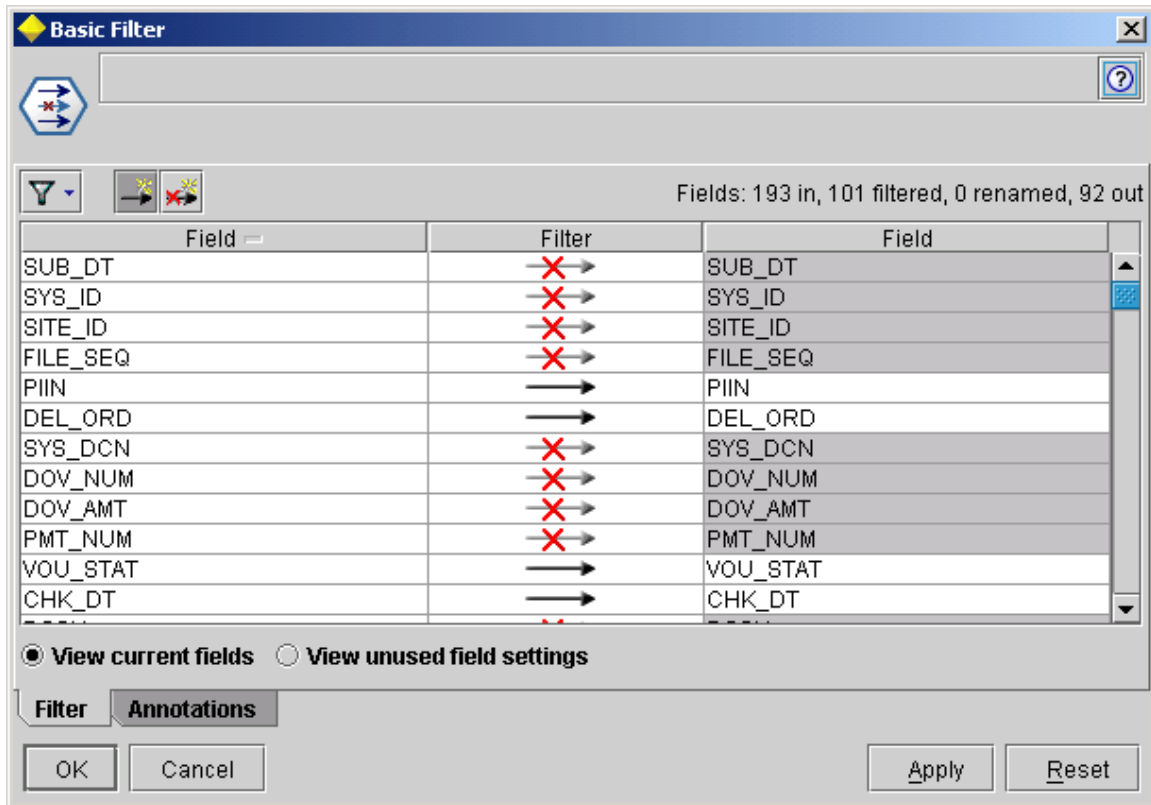


Figure 28 Basic Filter Node Dialog Box

Figure 28 is an example of a Filter Node dialog, where the modeler can remove or rename fields from the modeling stream. The other Filter Node dialogs in this Supernode are very similar so are not shown.

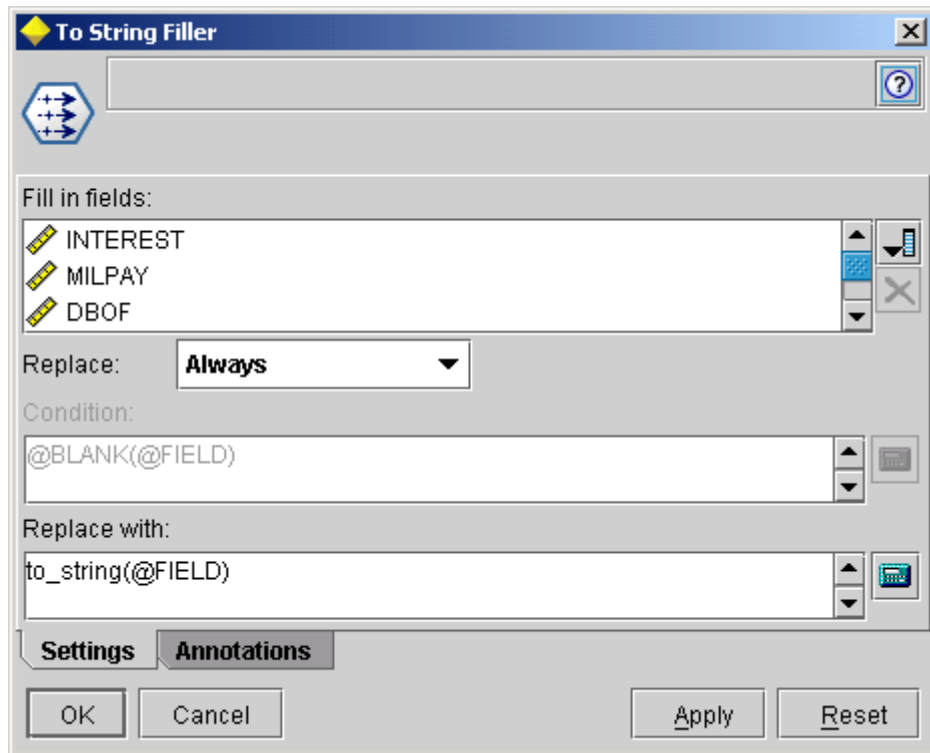


Figure 29 To String Filler Node Dialog Box

Figure 29 is an example of a Filler Node dialog. This particular one converts all fields to String storage prior to the Basic Type Node to prevent modeling problems downstream. The PMT_METH and PMT_TYPE Filler Nodes are similar, used to replace \$null\$ values with a new Set value, "Blank."

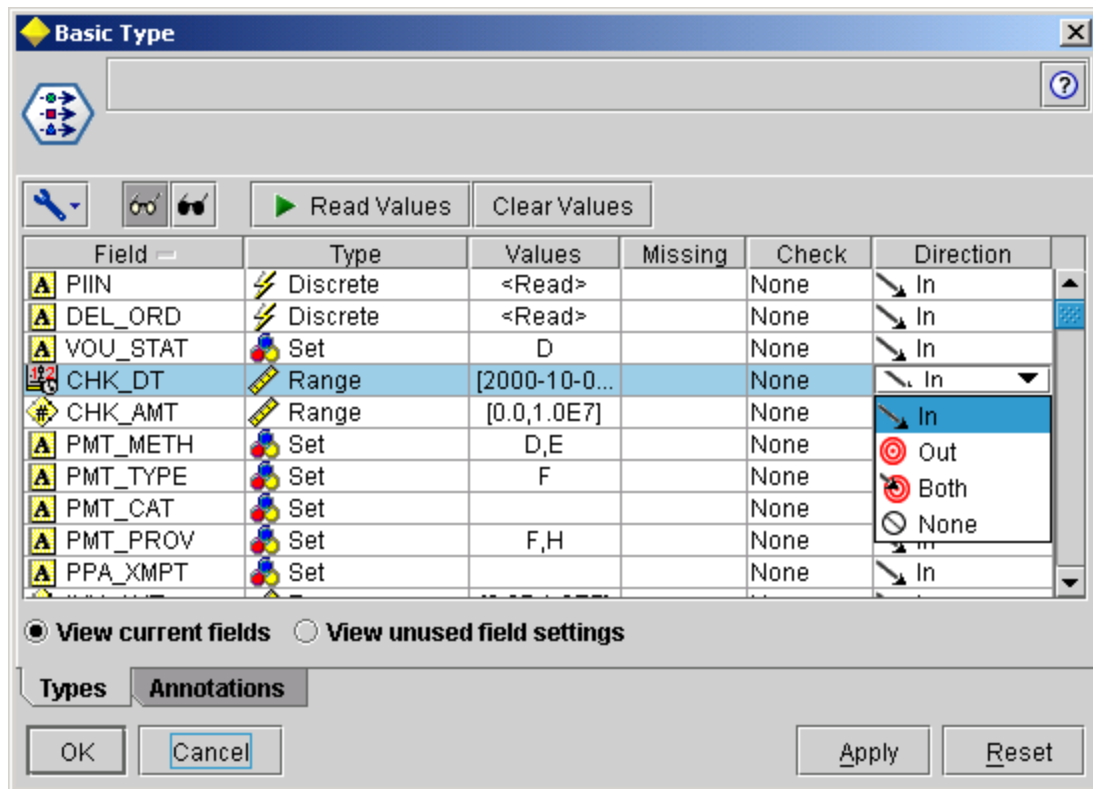


Figure 30 Basic Type Node Dialog Box

Figure 30 is an example of a Type Node, which specifies the variable storage, Type, and Values of each field, as well as the “Direction” of the field for modeling. The four possible Direction settings are In (used as an input or independent variable for modeling), Out (used as an response or dependent variable for modeling), Both (input and response), and None (not used for modeling). In this node all fields are initially set to “In.” The Final Type Node sets the Direction of all fields marked “A” in Appendix A to “None.” All other fields remain as “In.”

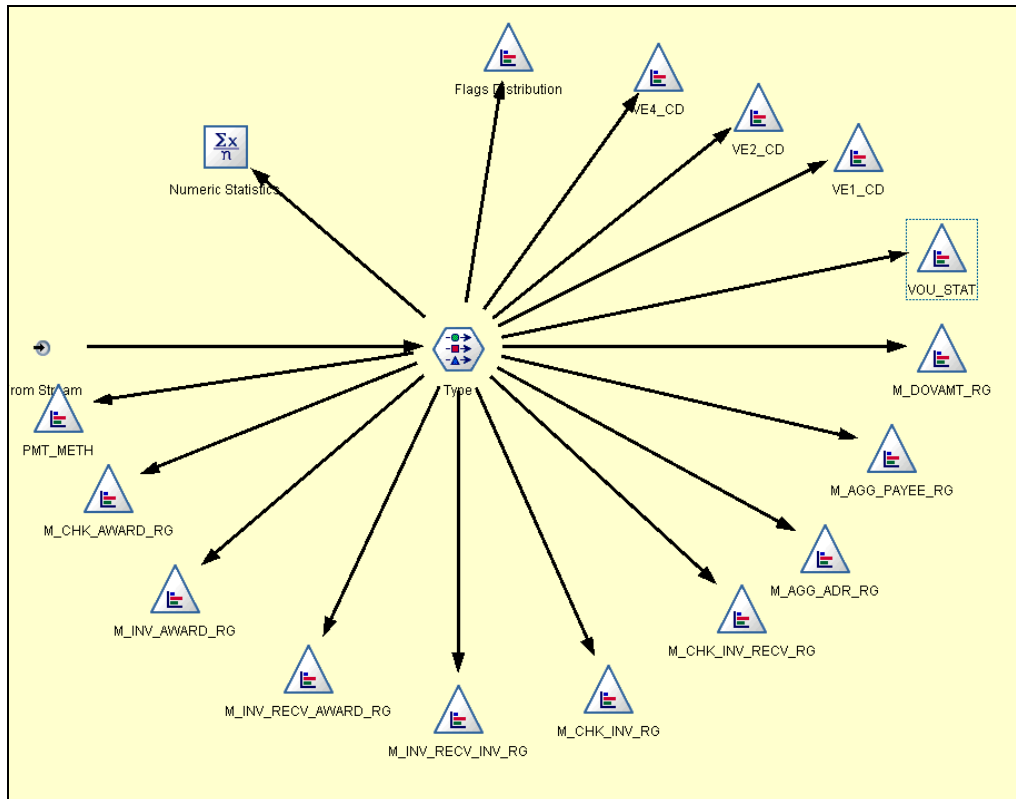


Figure 31 Distributions and Statistics Supernode

The Distributions and Statistics Supernode creates plots of the distribution of categorical variables as shown in Figure 33. This plot allows the modeler to determine the quality of categorical fields, which might contain all one value, for example, and thus be useless and inputs for modeling. Figure 32 is an example of a Distribution Node Dialog Box. All the other Distribution Nodes in this Supernode and their outputs are similar.

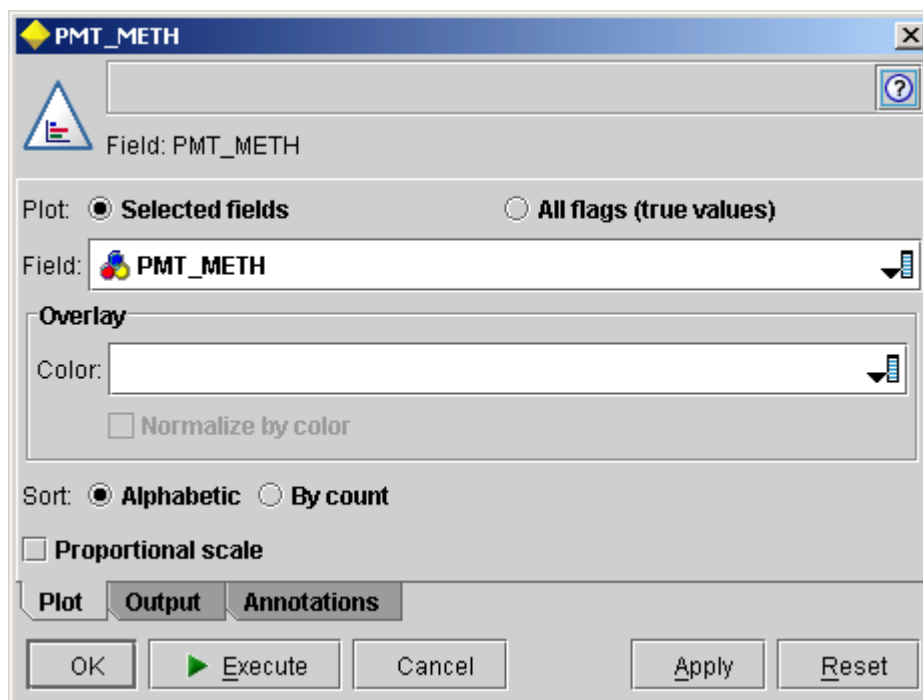


Figure 32 PMT_METH Distribution Node Dialog Box

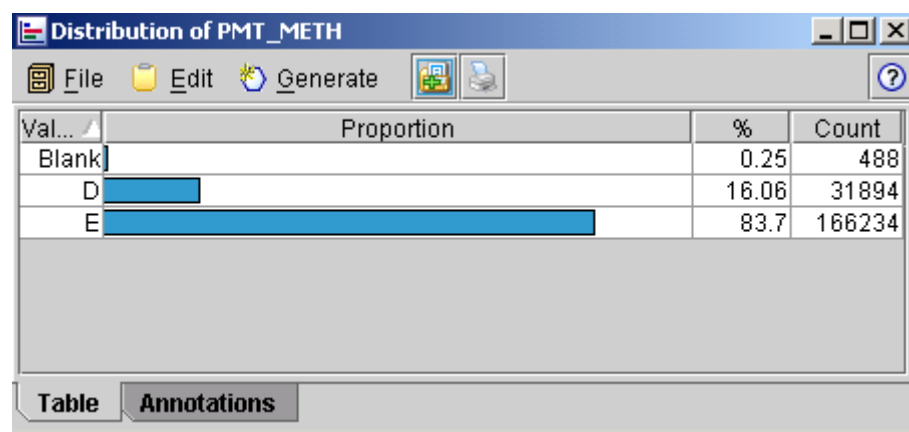


Figure 33 PMT_METH Distribution Plot

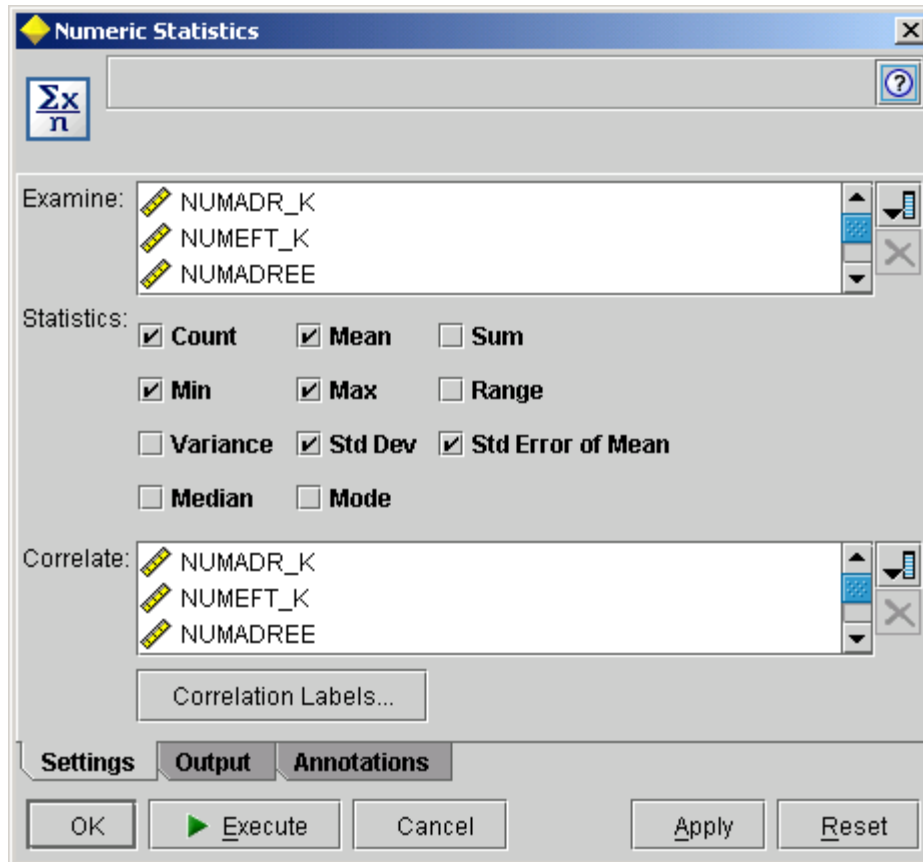


Figure 34 Numeric Statistics Node Dialog Box

As another check on quality of numeric (Range Type) fields, the Numeric Statistics Node (shown in Figure 34) produces output as shown in Figure 35, showing the modeler various statistics and correlation information about all these fields.

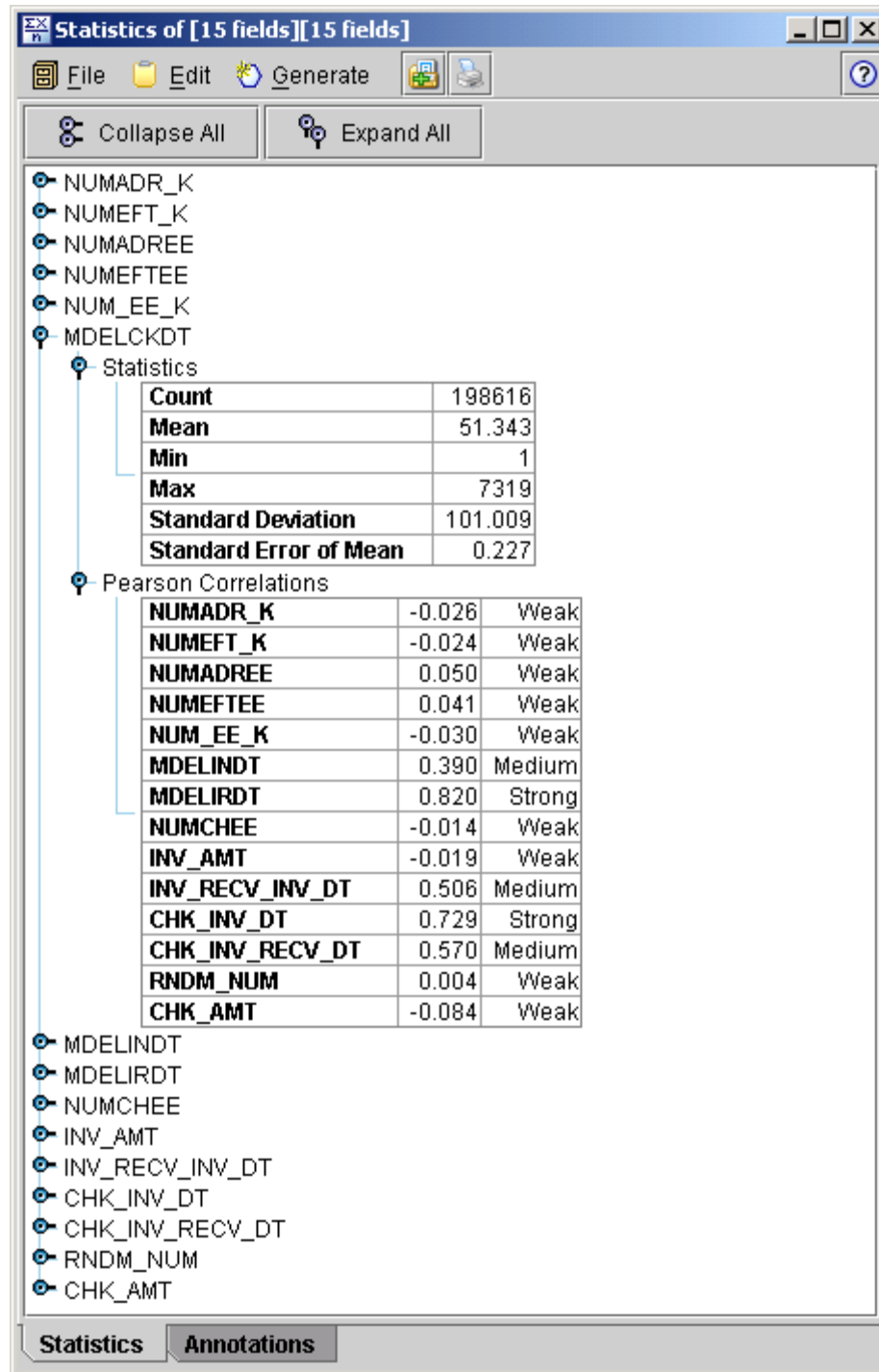


Figure 35 Numeric Statistics Node Output

The two Derive Nodes, shown in Figure 36 and Figure 37, are used to add new fields to the data stream for model

analysis. The ValSet Derive Node assigns Validation Set membership of A or B depending on the value of RNDM_NUM, a random number field generated in Access that comes from the database. Its purpose is to allow the modeler to divide the data set into two equally-sized random subsets for A/B validation of K-Means and Two Step cluster models. The Contract derive node creates a new field to identify specific contracts and enable analysis of clustering results with regard to contract distributions.

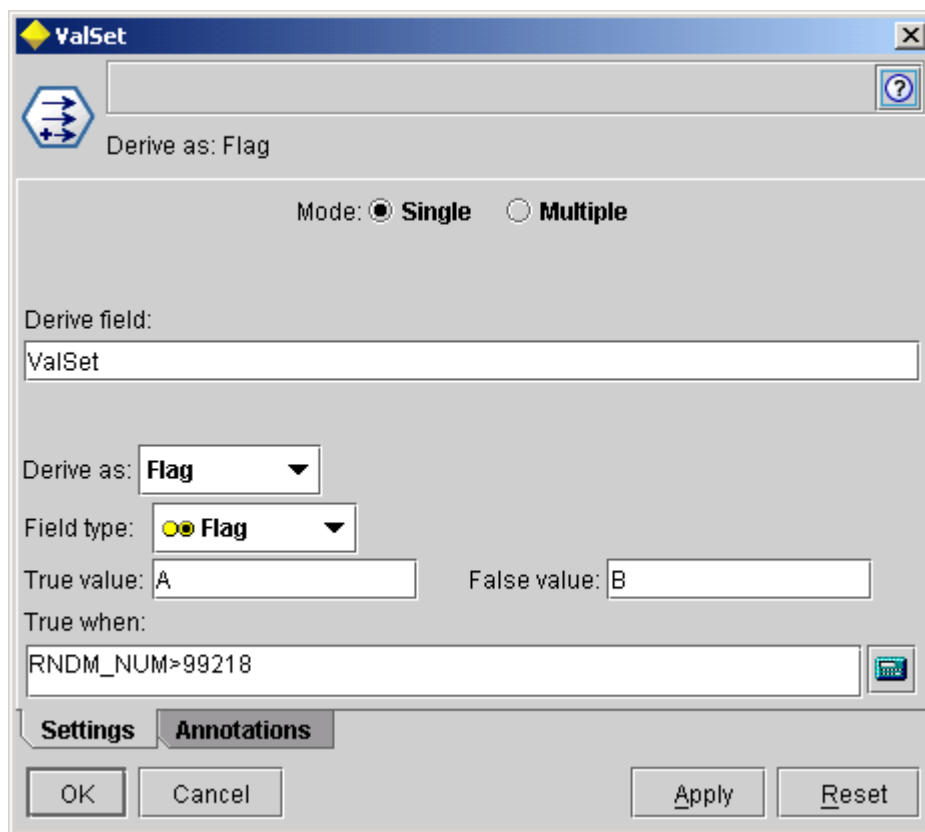


Figure 36 ValSet Derive Node Dialog Box



Figure 37 Contract Derive Node Dialog Box

2. KMEANS_UNSUP_POP_GWR MODELING STREAM

The purpose of this stream is to construct K-Means cluster models and produce output to be used to select the appropriate number of clusters and validate generated models.

a. Implementation

All K-Means models are built using the K-Means Model Dialog Box, shown in Figure 38. No Expert Options were selected. Figure 39 shows the PCA Modeling Dialog Box,

which was used to create the Principal Components Analysis model for the AB30 series clustering.

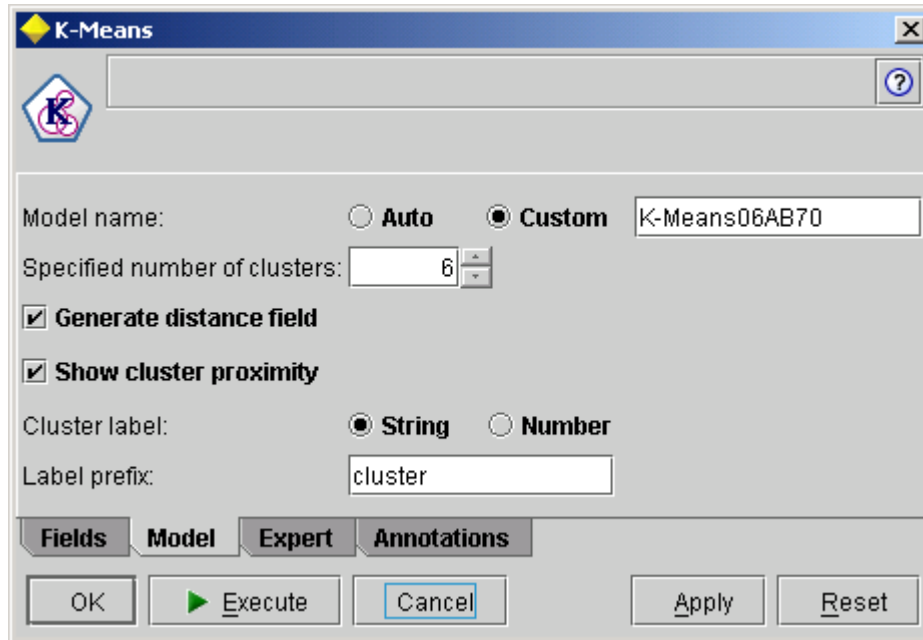


Figure 38 K-Means Model Node Dialog Box

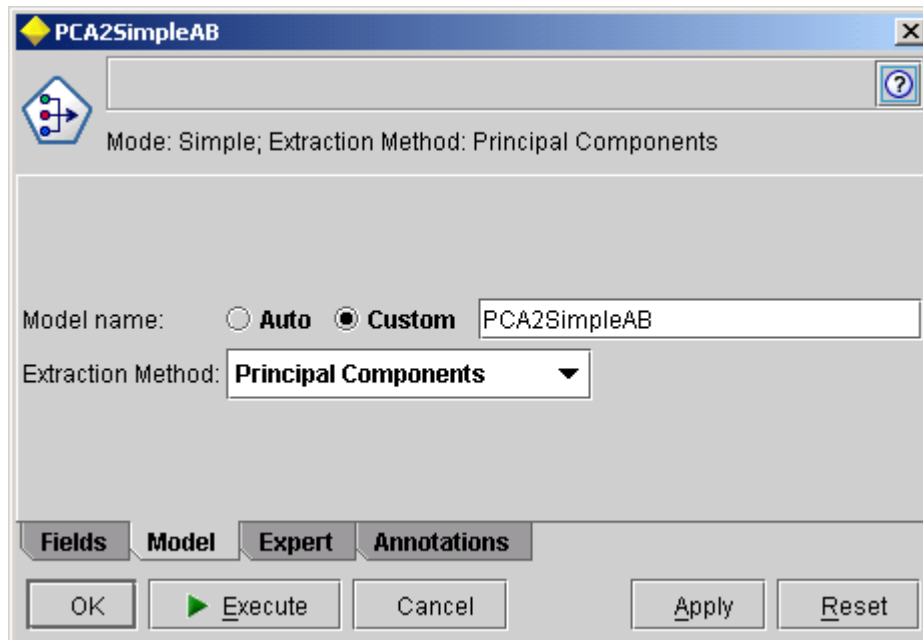


Figure 39 PCA Model Node Dialog Box

Figure 40 shows the end result of creating the multiple K-Means models required to evaluate the optimum number of clusters using the Sum of Squares method described in Chapter V. The AB20 Models Supernode and the AB30 PCA Models Supernode are both similar to the one shown.

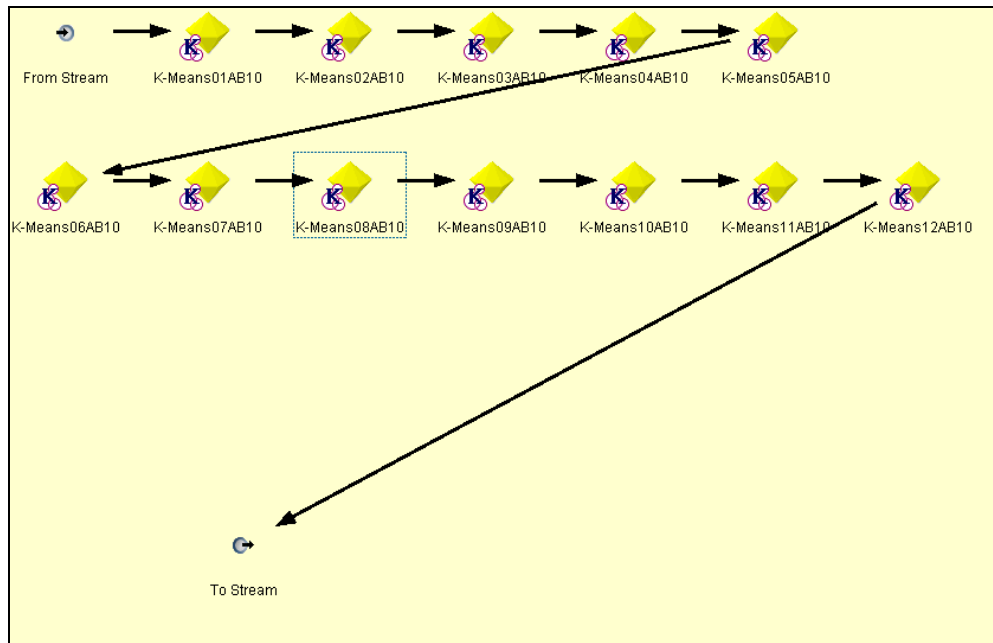


Figure 40 AB10 Models Supernode

Figure 41 illustrates the use of the field ValSet to select the validation subset which is used for modeling. Figure 42 shows how the data is passed through the two validation models built on validation sets A and B.

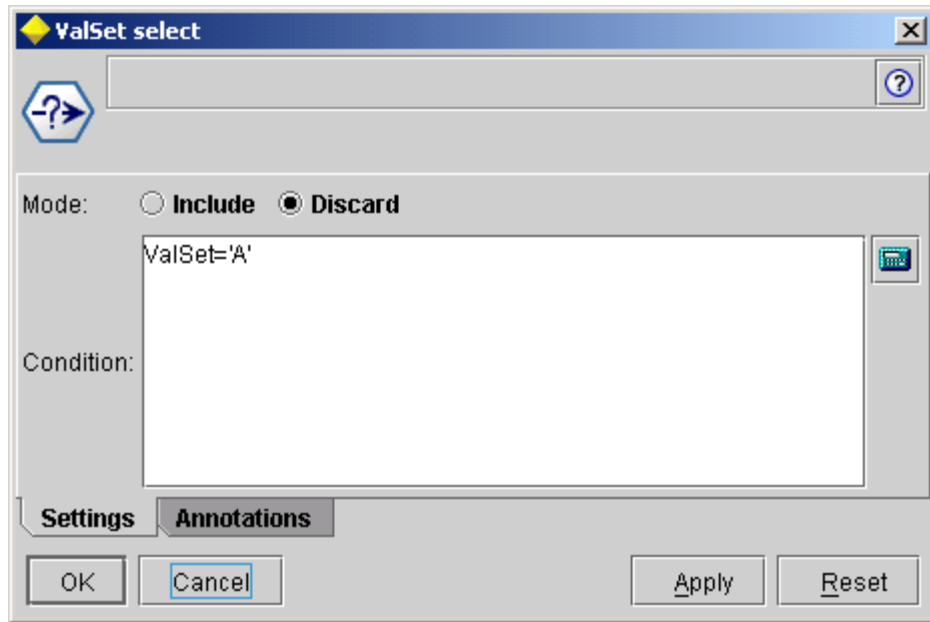


Figure 41 ValSet Select Node (currently selects ValSet "B")

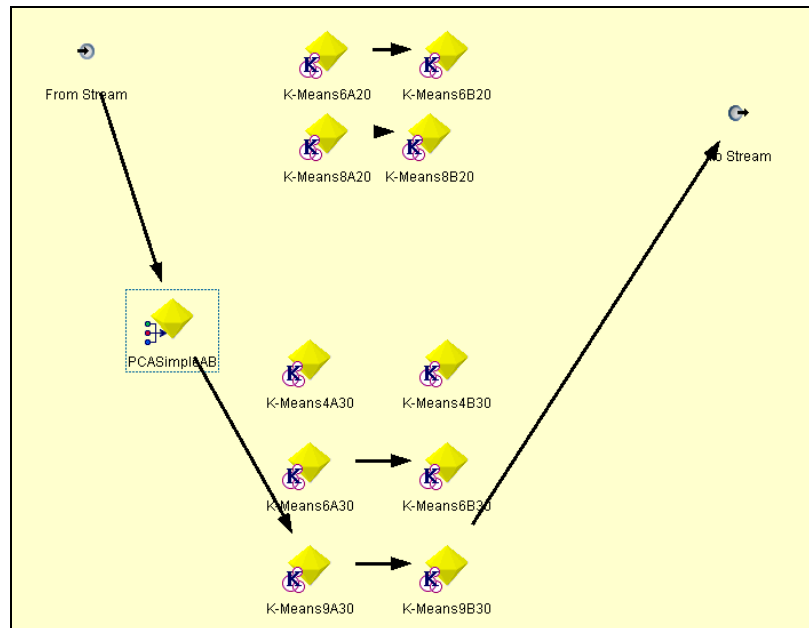


Figure 42 A/B Validation Models Supernode

Figure 43 is an example of a Matrix Node Dialog Box, which is used to generate cross-tabulation of cluster assignments for the A/B validation process.

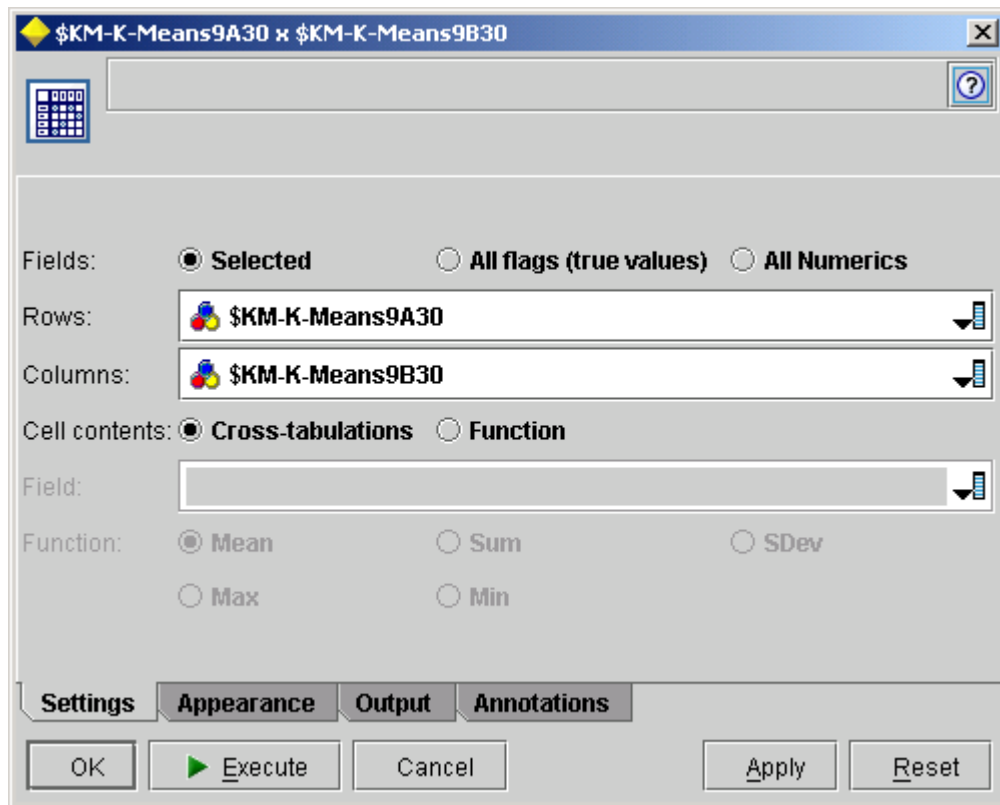


Figure 43 K-Means A30/B30 Matrix Dialog Box

The Sum of Squares Supernode (Figure 44) produces a table of within-cluster sum of squares of the distance fields for each model contained in one of the AB Models Supernodes.

The `_Square` Node (Figure 45) creates a field containing the square of the distance field `$KMD-<Model Name>` for each record. The Within-Cluster Sum of Squares Set Globals Node (Figure 46) sums these squared values for each model and creates Global fields for each value. This is the desired result, and it would be possible to stop at this point. However, the following sequence of nodes produces data in a format that is much easier to use in producing a W_k vs. k graph such as shown in Chapter V.

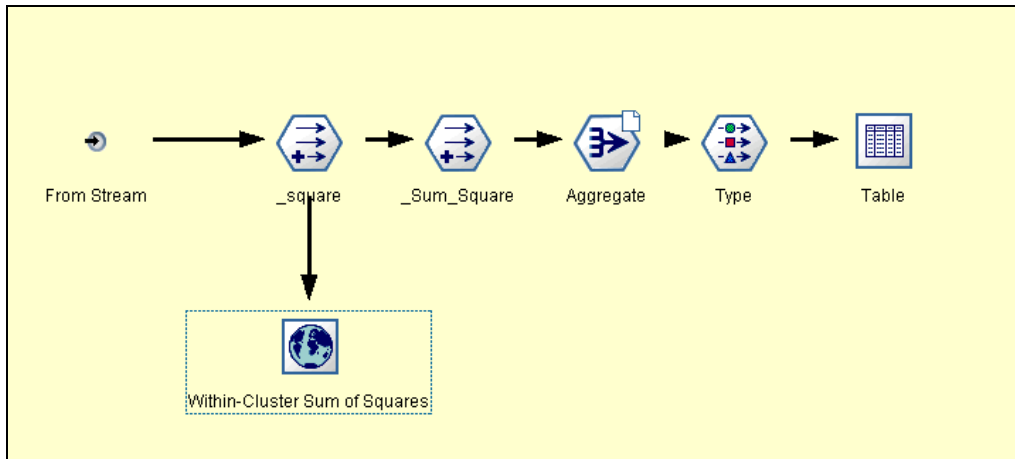


Figure 44 Sum of Squares Supernode

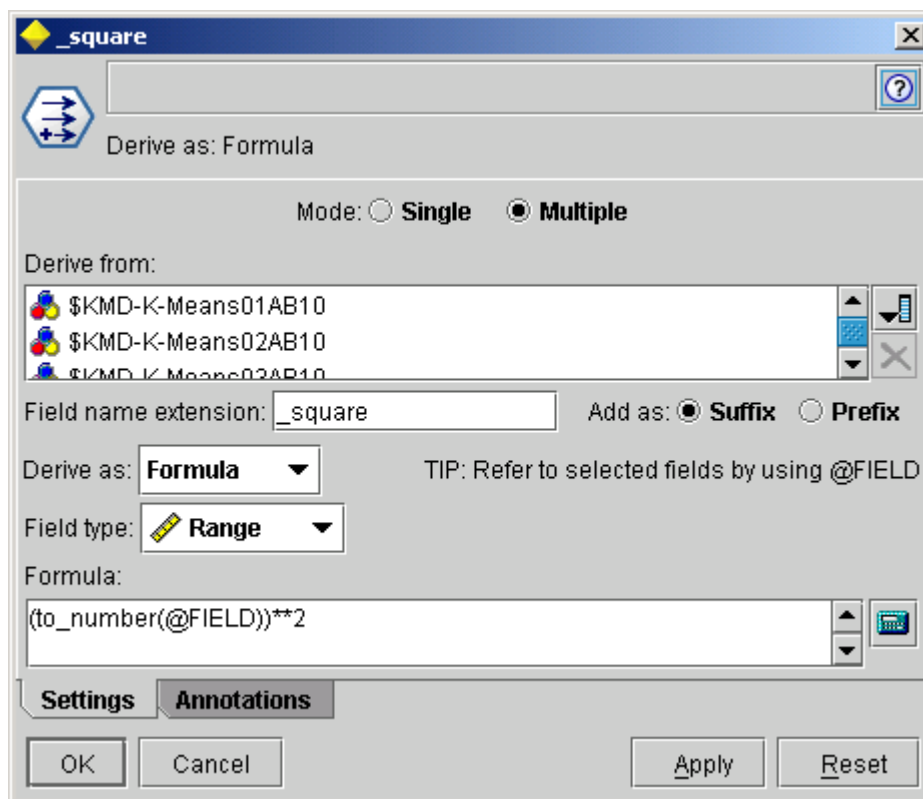


Figure 45 _Square Derive Node Dialog Box

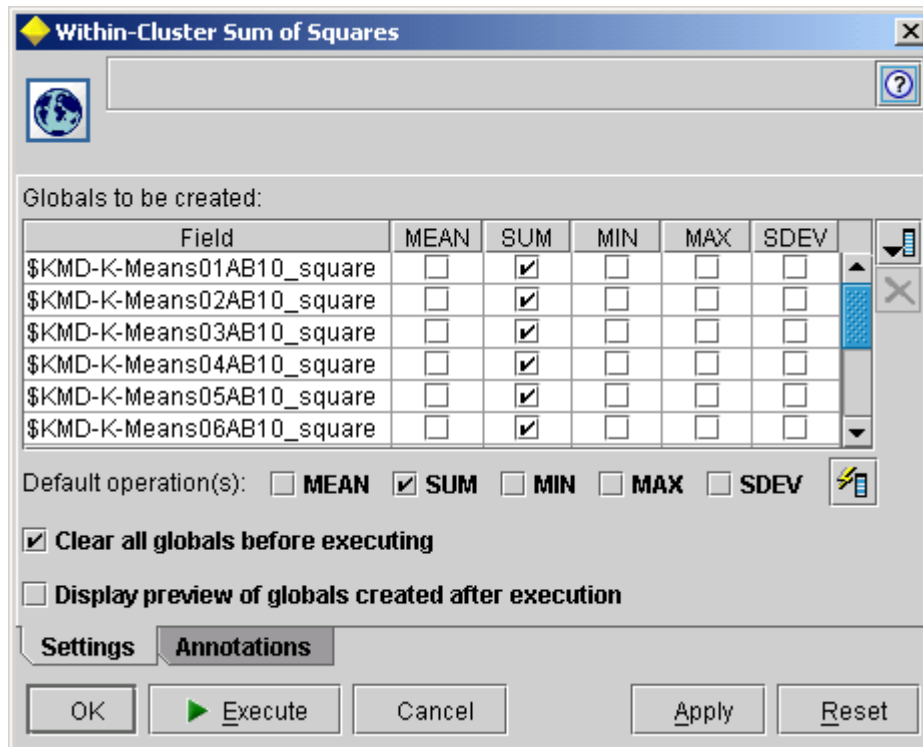


Figure 46 Within-Cluster Sum of Squares Set Globals Dialog Box

The `_Sum_Square` Derive Node (Figure 47) creates another field and assigns it the Global value that was created by the Set Globals Node described above. These sums of squares are then compiled by the Aggregate Node shown in Figure 48, and this aggregation is sent through a Type Node to the Table Node which produces useable output. This table can be copied and pasted into Excel for easy production of a graph of Within-Cluster Sum of Squares vs. Number of Clusters to be used to evaluate the proper number of clusters for a particular model.

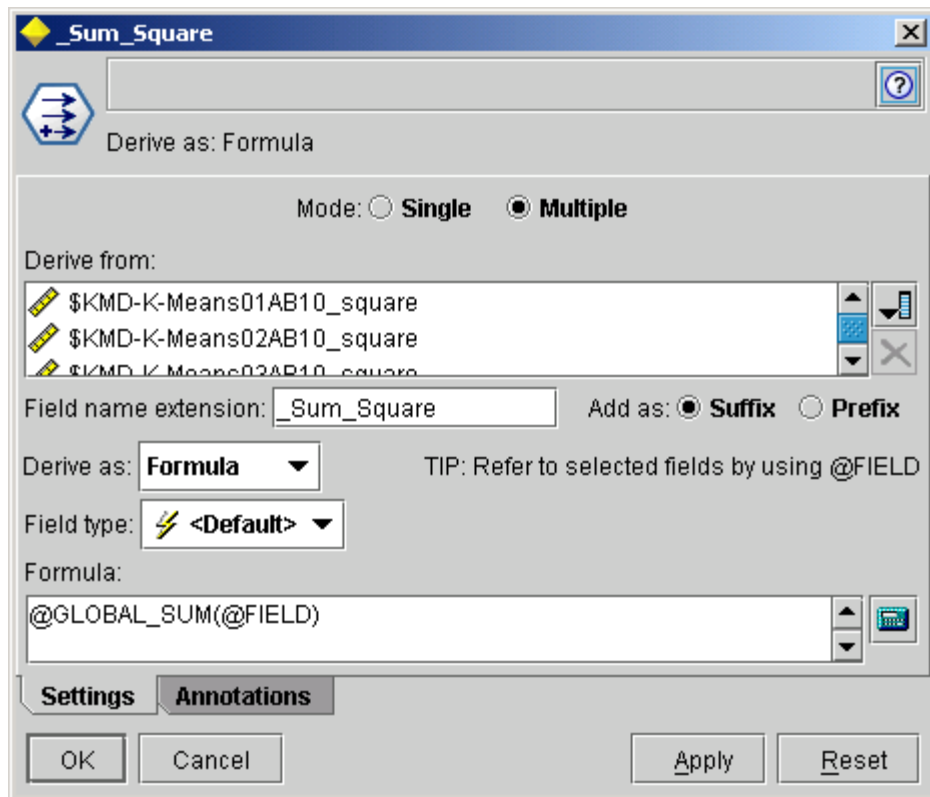


Figure 47 _Sum_Square Derive Node Dialog Box

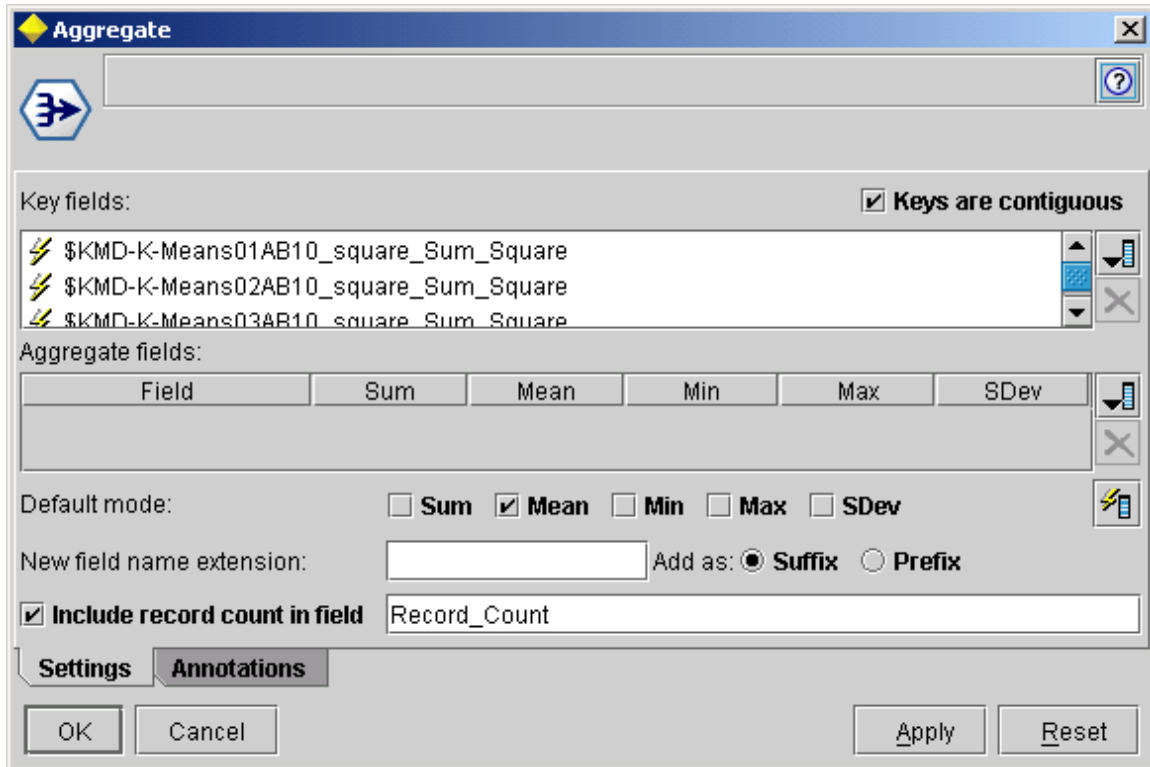


Figure 48 Sum of Squares Aggregate Node Dialog Box

b. Results

Table 13 shows the Principal Components extraction of the numeric data used for the AB30 series of cluster models.

Component Matrix(a)					
	Component				
	1	2	3	4	5
CHK_AMT	-.169	-.170	.329	-.370	.710
INV_AMT	-6.161E-02	-8.265E-02	.170	-.140	.535
INV_RECV_INV_DT	.767	4.962E-02	-.299	-.211	.116
CHK_INV_DT	.908	6.193E-02	-.346	-.158	.142
CHK_INV_RECV_DT	.526	3.950E-02	-.191	1.189E-02	8.816E-02
NUMADR_K	-.101	.943	4.190E-02	4.378E-02	.171
NUMEFT_K	-7.134E-02	.534	1.287E-02	1.214E-02	3.304E-02
NUMADREE	9.603E-02	-.129	-.101	.823	.323
NUMEFTEE	8.529E-02	-.123	-.105	.797	.396
NUM_EE_K	-.105	.951	3.900E-02	3.949E-02	.159
MDELCKDT	.922	6.323E-02	.334	3.592E-02	-3.029E-02
MDELINDT	4.129E-02	3.345E-03	.924	.262	-.233
MDELIRDT	.755	4.946E-02	.540	3.544E-02	-9.825E-02
NUMCHEE	-6.120E-02	-.186	.250	-.320	.458
Extraction Method: Principal Component Analysis.					
a 5 components extracted.					

Table 13 PCA Factor Analysis Component Matrix

The K-Means model selected for use in further analysis was K-Means06AB20, built with six clusters on the numeric fields only. Figure 50 and Figure 49 show the input fields and cluster distribution, respectively, for this model. Table 14 shows the co-clustering matrix for the A and B validation models, resulting in Cramer's Coefficient = 83.23%.

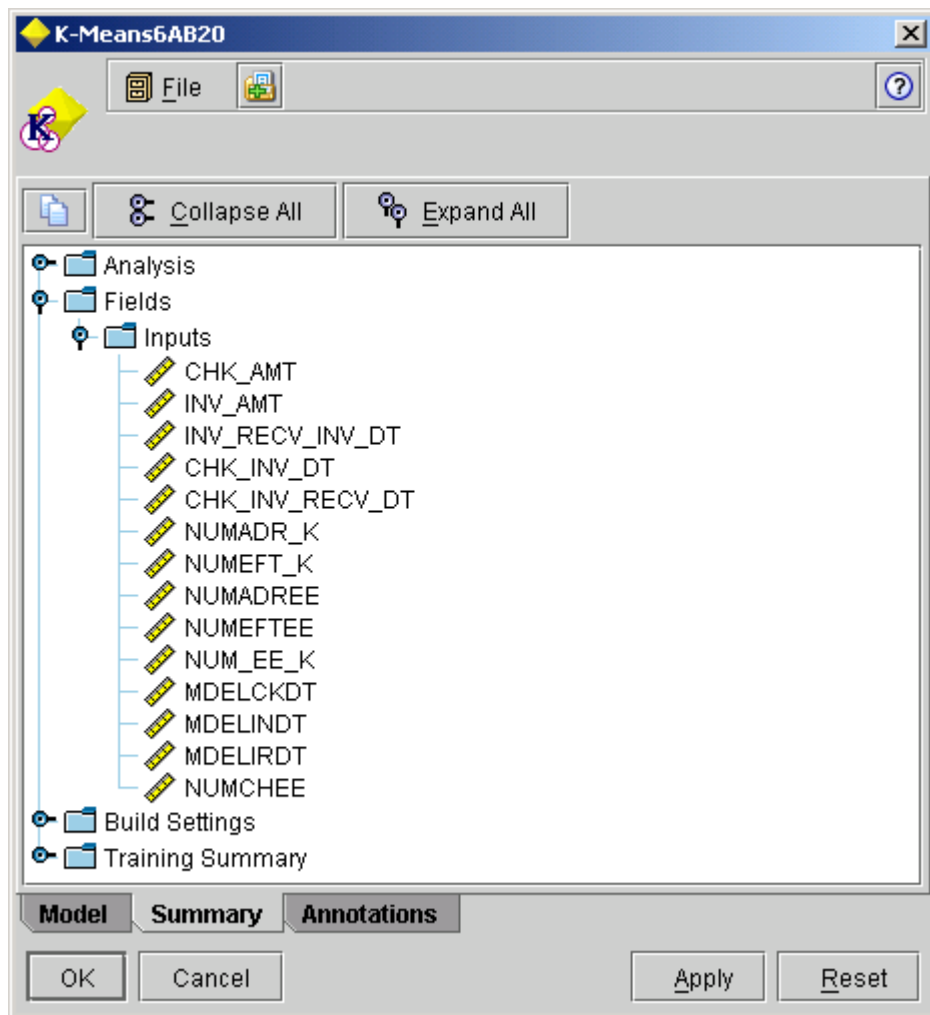


Figure 49 K-Means06AB20 Generated Model Node, Summary Tab

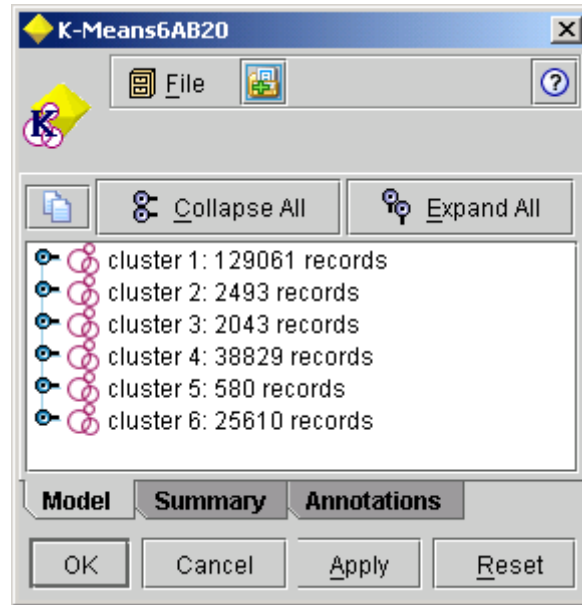


Figure 50 K-Means06AB20 Generated Model Node, Model Tab

K-Means06B20							
K-Means06A20		cluster-1	cluster-2	cluster-3	cluster-4	cluster-5	cluster-6
	cluster-1	63558	0	0	0	0	11292
	cluster-2	957	1007	0	252	0	55
	cluster-3	0	268	0	0	0	0
	cluster-4	19	0	0	19166	0	1419
	cluster-5	0	0	0	0	283	0
	cluster-6	0	0	1032	0	0	0

Table 14 A/B Validation Matrix for K-Means06AB20

The following sequence of figures and tables illustrates the effect of changing the order of data for a K-Means clustering model, comparing the results of using numeric fields only to using categorical fields only. The model K-Means06AB50 was generated using numeric fields only, with the data sorted by RNDM_NUM. Figure 51 shows the cluster distribution of this model, and Table 15 shows the cross-tabulation of cluster assignments, resulting in Cramer's Coefficient = 83.03%. Figure 52, Figure 53, and

Table 16 show the same process for categorical fields only, resulting in Cramer's Coefficient = 72.12%.

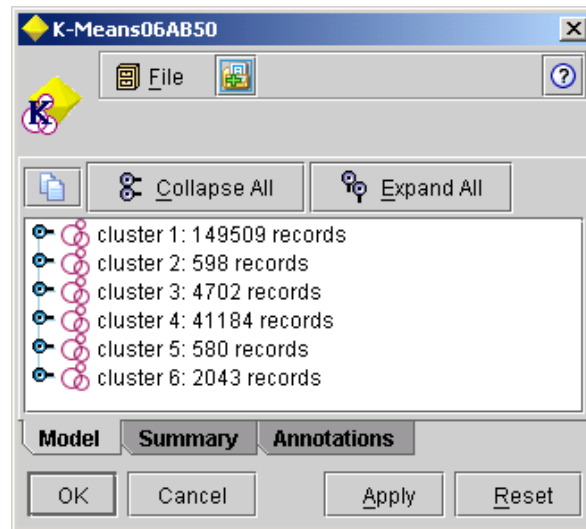


Figure 51 K-Means06AB50 Generated Model Node, Model Tab

	cluster-1	cluster-2	cluster-3	cluster-4	cluster-5	cluster-6
cluster-1	126885	0	2136	38	0	0
cluster-2	4	598	1890	3	0	0
cluster-3	0	0	0	0	0	2043
cluster-4	0	0	536	38293	0	0
cluster-5	0	0	0	0	580	0
cluster-6	22628	0	133	2849	0	0

Table 15 Cross-Tabulation of Cluster Assignment, K-Means06AB20 vs. K-Means06AB50 Models

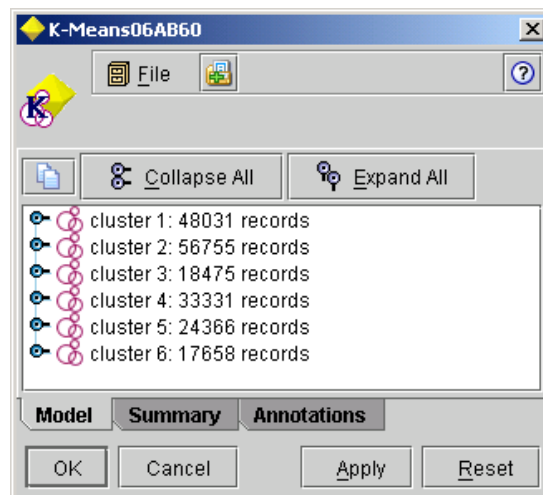


Figure 52 K-Means06AB60 Generated Model Node, Model Tab

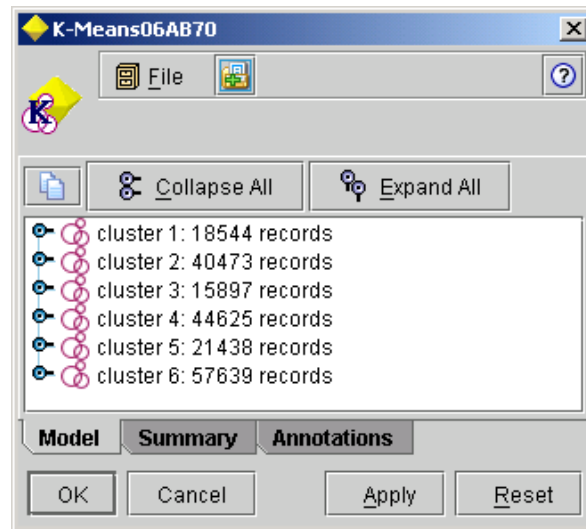


Figure 53 K-Means06AB70 Generated Model Node, Model Tab

	cluster-1	cluster-2	cluster-3	cluster-4	cluster-5	cluster-6
cluster-1	4	350	99	14147	6	31886
cluster-2	5	15552	12746	24739	4407	337
cluster-3	18404	31	9	32	0	0
cluster-4	124	24273	2799	5156	0	1771
cluster-5	1	224	112	5	272	23765
cluster-6	6	71	65	692	16526	0

Table 16 Cross-Tabulation of Cluster Assignment, K-Means06AB60 vs. K-Means06AB70 Models

3. TWOSTEP_UNSUP_POP_GWR MODELING STREAM

Figure 54 shows the dialog used to create the Two Step clustering model used for analysis. Figure 55 shows the cluster distribution for the generated model, and Table 17 shows the co-clustering matrix for the A and B validation models, resulting in Cramer's Coefficient = 91.96%.

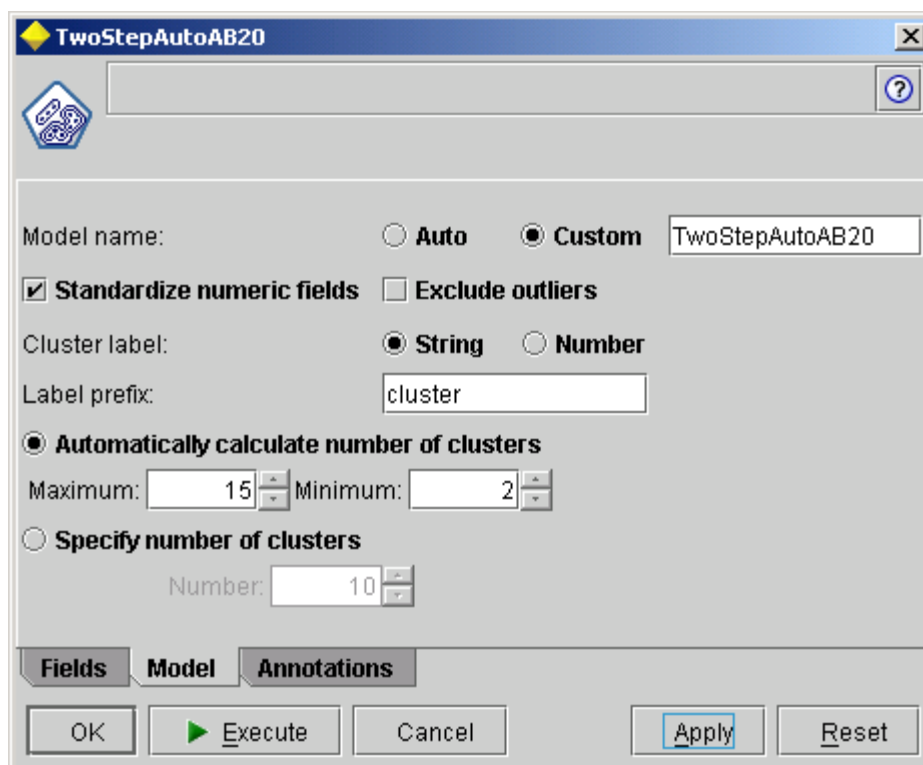


Figure 54 TwoStep Model Node Dialog Box

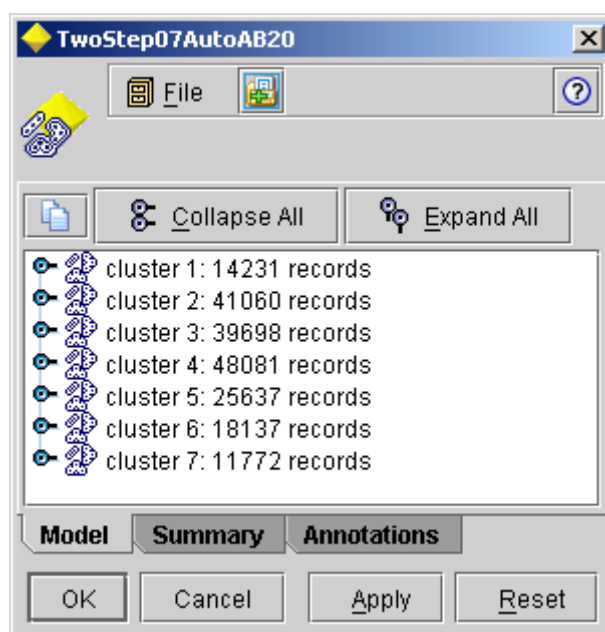


Figure 55 TwoStep07AutoAB20 Generated Models Dialog Box, Model Tab

	TwoStep07B20						
TwoStep07A20	cluster-1	cluster-2	cluster-3	cluster-4	cluster-5	cluster-6	cluster-7
cluster-1	7477	33	0	0	0	0	0
cluster-2	0	0	0	0	17	0	5022
cluster-3	0	139	990	504	11080	0	99
cluster-4	10	443	19686	1800	401	0	0
cluster-5	0	10	34	16987	5044	0	2
cluster-6	0	0	1	0	1	9058	0
cluster-7	190	20157	54	0	69	0	0

Table 17 A/B Validation Matrix for TwoStep07AutoAB20

4. KOHONEN_UNSUP_POP_GWR MODELING STREAM

Figure 56 and Figure 57 show the Model Node settings used in building the KSOM10x11AB02 model used for analysis. The only Expert settings used were to adjust the dimensions of the Kohonen map. Figure 58 is used to create the two-dimensional plot of the Kohonen prototypes and record assignments for evaluation. Figure 59 shows the input and output layers for the generated model.

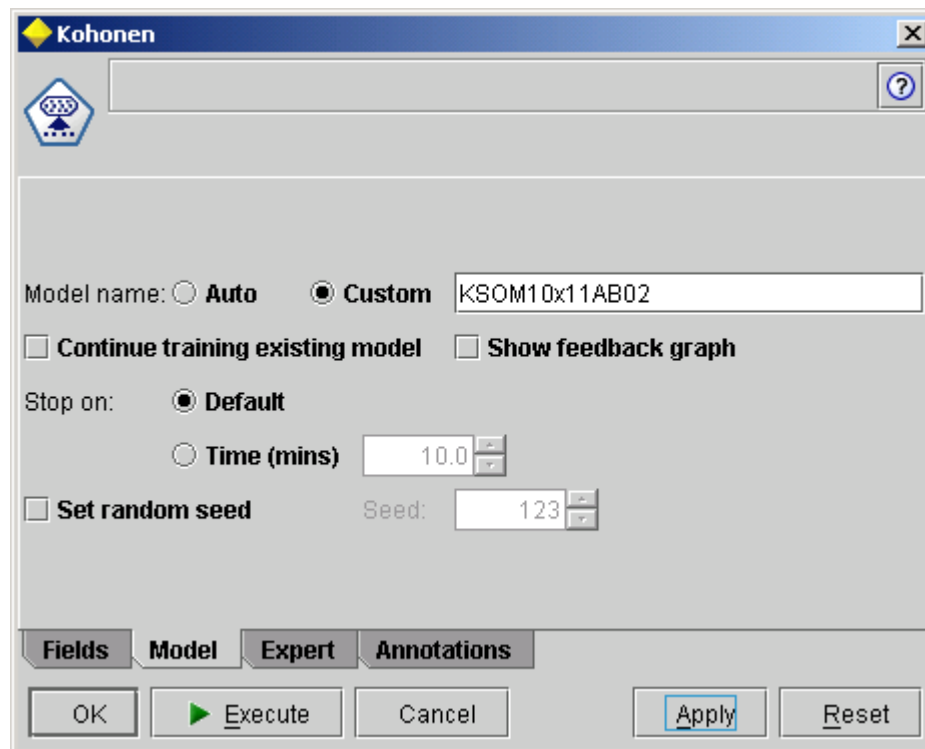


Figure 56 Kohonen Model Node Dialog Box, Model Tab

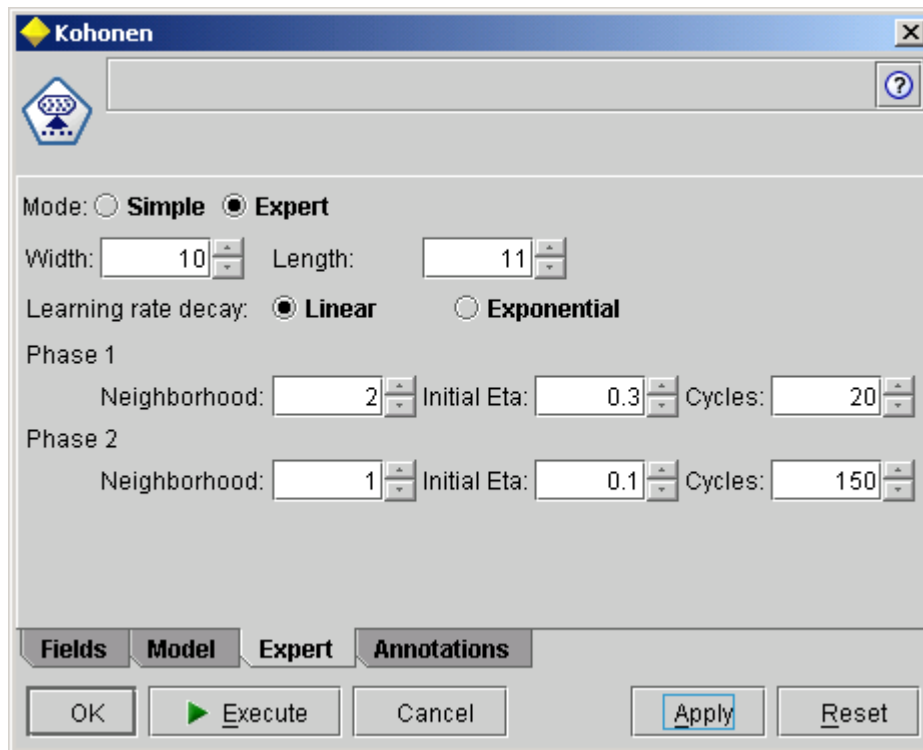


Figure 57 Kohonen Model Node Dialog Box, Expert Tab

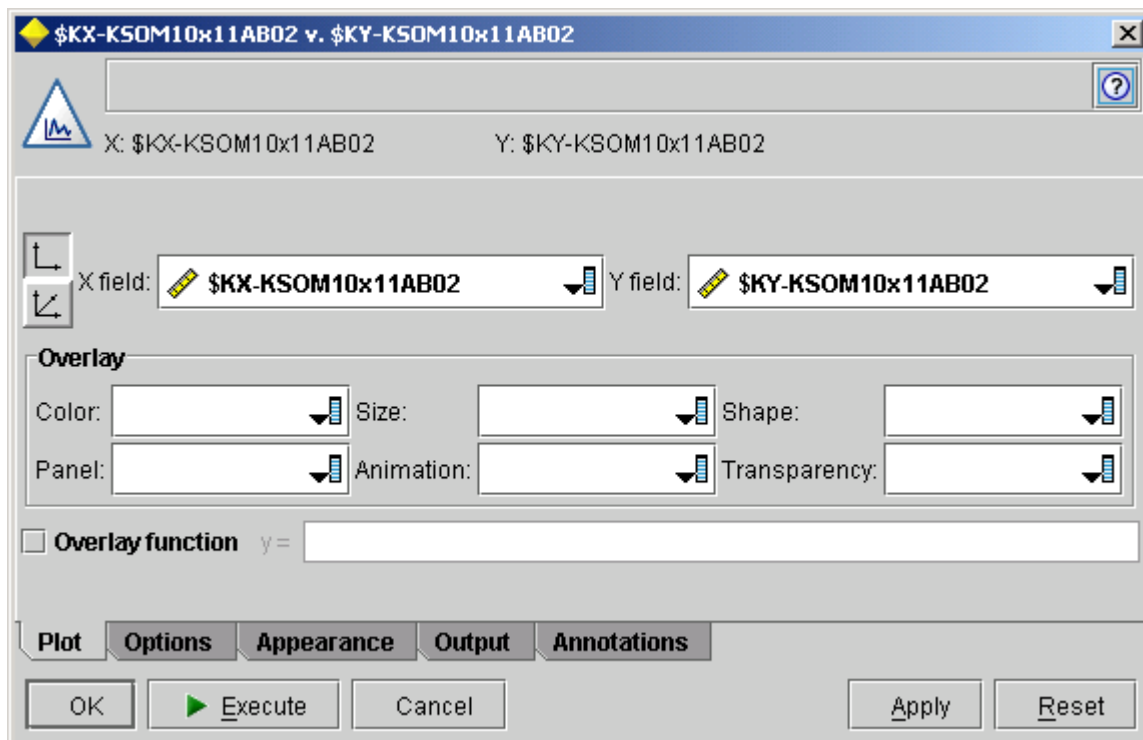


Figure 58 Kohonen Model Plot Dialog Box

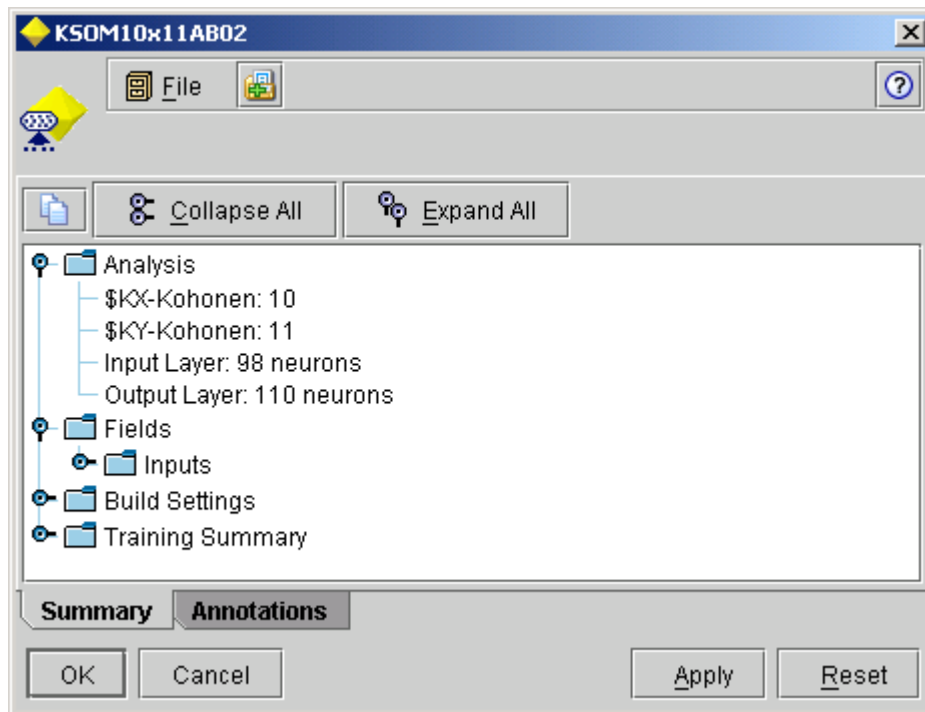


Figure 59 Kohonen Generated Model Dialog Box, Summary Tab
5. MODEL_ANALYSIS_POP_GWR ANALYSIS STREAM

a. Implementation

The model analysis stream produces a table of transactions that have been identified as orphans in all three of the generated models. There is also the option to identify sparse prototypes in the Kohonen map, accomplished by the Sparse Prototypes Supernode (Figure 60). Figure 61 shows the aggregation on Kohonen prototype fields, which after sorting produces a table showing each prototype and the number of records it contains (Figure 62 is an example). This table is used to identify the sparse prototypes.

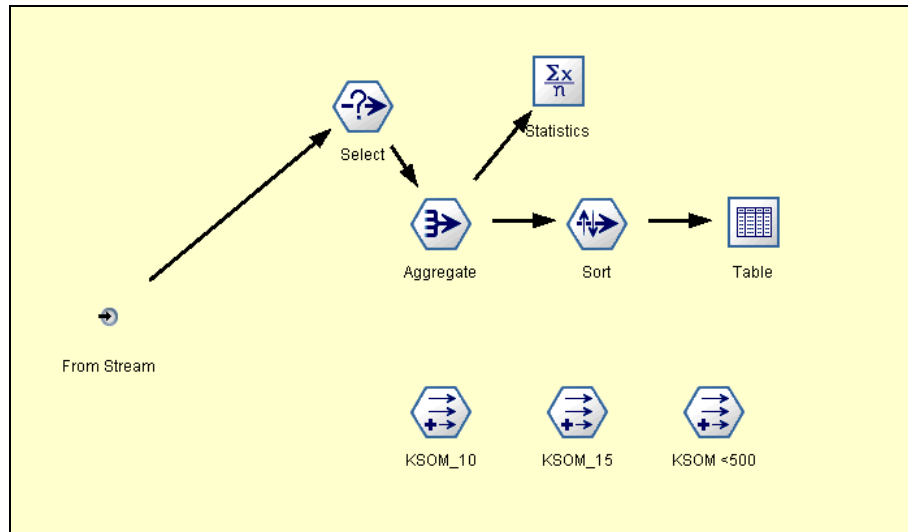


Figure 60 Sparse Prototypes Supernode

Aggregate

Key fields: ☐ **Keys are contiguous**

- \$KX-KSOM10x11AB02
- \$KY-KSOM10x11AB02

Aggregate fields:

Field	Sum	Mean	Min	Max	SDev

Default mode: ☒ **Sum** ☒ **Mean** ☐ **Min** ☐ **Max** ☐ **SDev**

New field name extension: Add as: ☒ **Suffix** ☐ **Prefix**

☒ **Include record count in field**

Settings **Annotations**

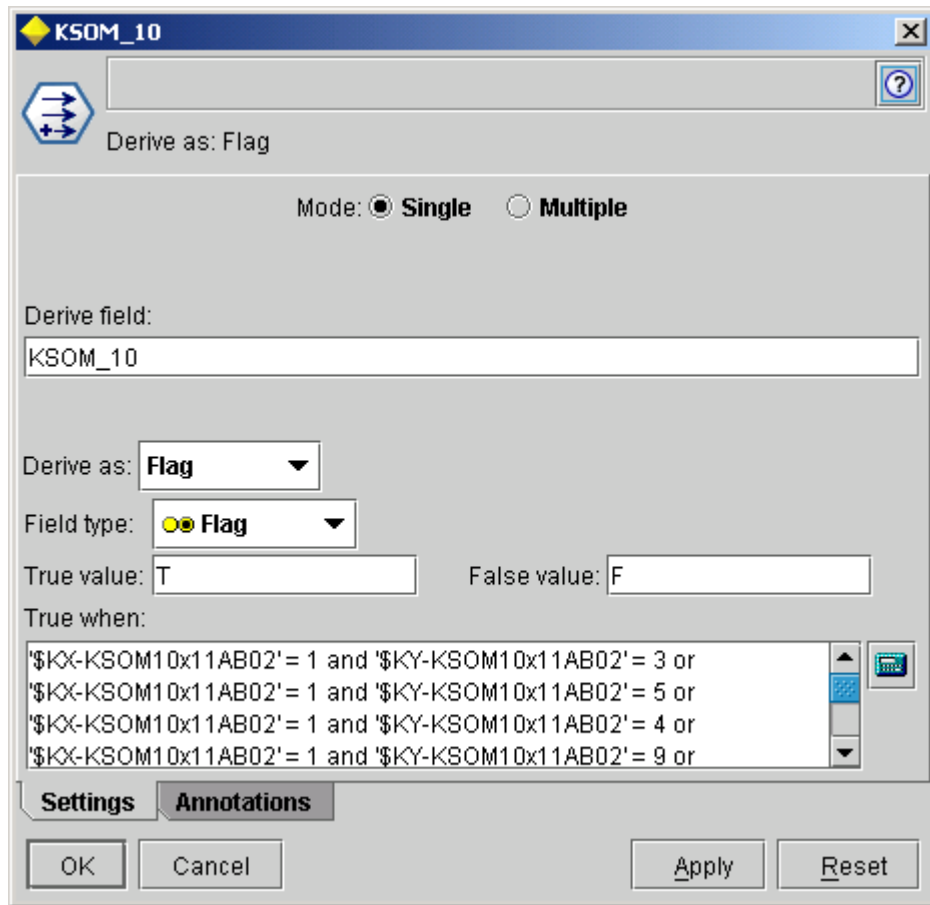
OK Cancel Apply Reset

Figure 61 Aggregate Node Settings

	\$KX-KSOM10x11AB02	\$KY-KSOM10x11AB02	Record_Count
98	3	6	230
99	6	1	198
100	8	5	175
101	1	3	167
102	1	5	164
103	1	4	155
104	1	9	139
105	9	1	131
106	5	0	116
107	2	6	114
108	7	1	102
109	8	1	85
110	8	3	81

Figure 62 Table of Kohonen Prototypes Sorted in Descending Order by Number of Transactions

After selection of the appropriate metric for determining a sparse prototype, a Derive Node can be generated from the generated table. An example is shown in Figure 63, which identifies records belonging to one of the ten sparsest nodes. The other two Derive Nodes in the Supernode perform the same function.



The image shows a software dialog box titled "KSOM_10". It has a standard Windows-style title bar with a close button. Below the title bar is a toolbar with a blue icon of three arrows pointing right and a question mark icon. The main area of the dialog is labeled "Derive as: Flag". Below this, there are two radio buttons for "Mode": "Single" (which is selected) and "Multiple". A text field labeled "Derive field:" contains the text "KSOM_10". Below that, there is a dropdown menu for "Derive as:" with "Flag" selected. Next to it is another dropdown for "Field type:" with "Flag" selected. Below these are two text fields: "True value:" with "T" and "False value:" with "F". A large text area labeled "True when:" contains a logical expression: `'$KX-KSOM10x11AB02'= 1 and '$KY-KSOM10x11AB02'= 3 or '$KX-KSOM10x11AB02'= 1 and '$KY-KSOM10x11AB02'= 5 or '$KX-KSOM10x11AB02'= 1 and '$KY-KSOM10x11AB02'= 4 or '$KX-KSOM10x11AB02'= 1 and '$KY-KSOM10x11AB02'= 9 or`. At the bottom, there are two tabs: "Settings" (active) and "Annotations". Below the tabs are four buttons: "OK", "Cancel", "Apply", and "Reset".

Figure 63 KSOM_10 Derive Node Settings

The Contract Count Supernode (Figure 64) produces a field containing the number of transactions in the contract to which each record belongs, which is essential to identifying orphan transactions.

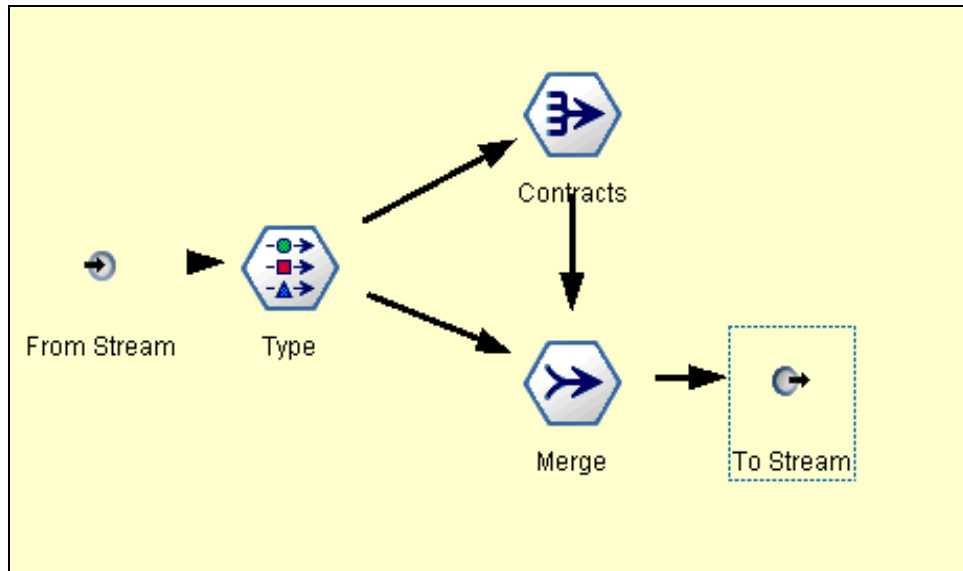


Figure 64 Contract Count Supernode

The data is first aggregated by contract, and then the Merge Node (Figure 65 and Figure 66) creates a new field with the number of contracts for each transaction.

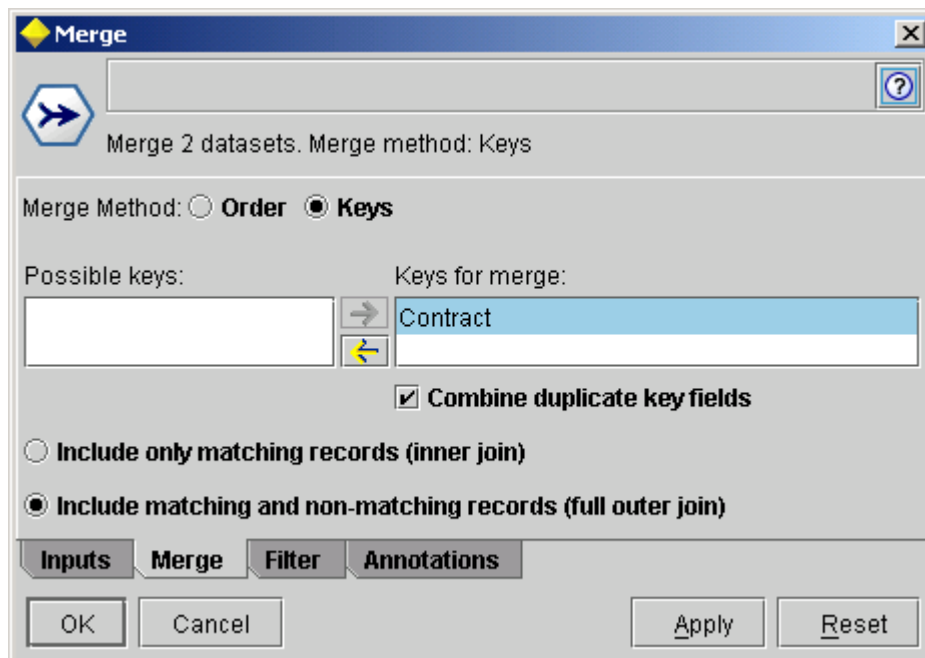


Figure 65 Merge Node Dialog Box, Merge Tab

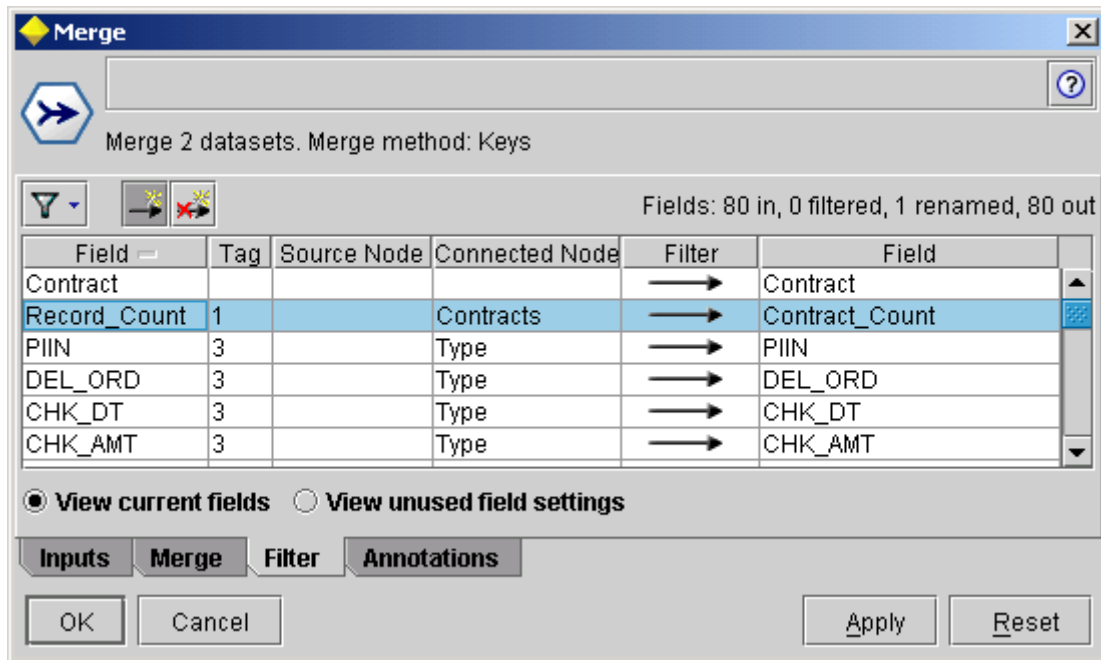


Figure 66 Merge Node Dialog Box, Filter Tab

The Orphans Supernode (Figure 67) accomplishes the important task of creating fields identifying records as orphans for one or more of the generated models. For each type of model, Two Step, K-Means, and Kohonen, the data is first merged on contract and cluster number (prototype number in the Kohonen case), then merged back to create a field identifying the number of transactions in each cluster from each contract. Figure 68 shows an example Merge Node Filter Tab, with the new field TS_Cluster_Count. The other two merge nodes are very similar and produce the new fields KM_Cluster_Count and KSOM_Prototype_Count.

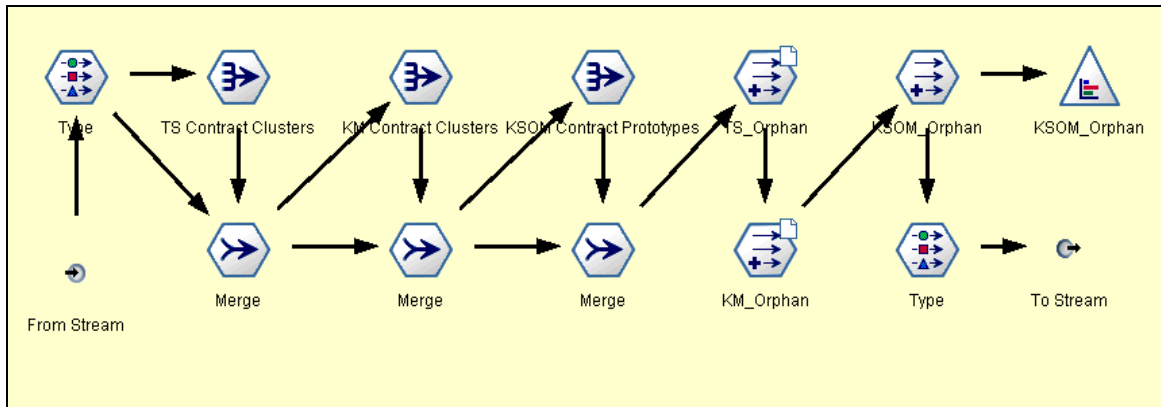


Figure 67 Orphans Supernode

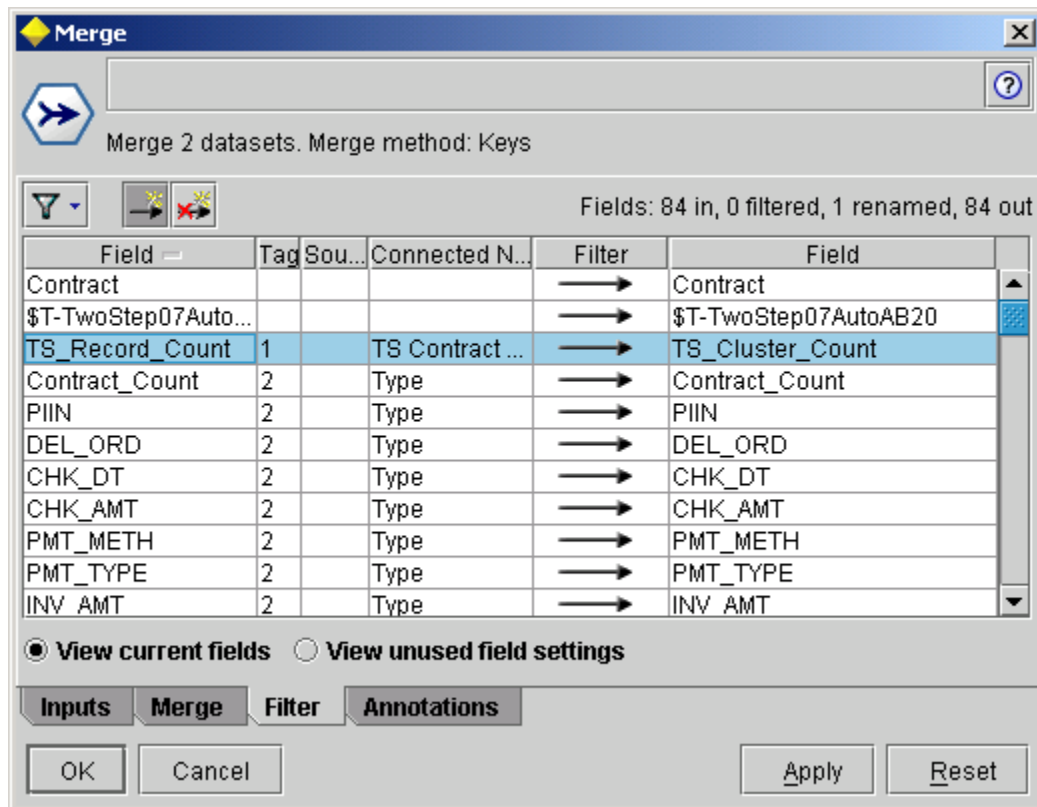


Figure 68 Merge Node Filter Settings

The three Derive Nodes create new Flag fields to identify orphan transactions. Figure 69 shows an example for the Two Step orphans; the other two derive nodes are very similar.

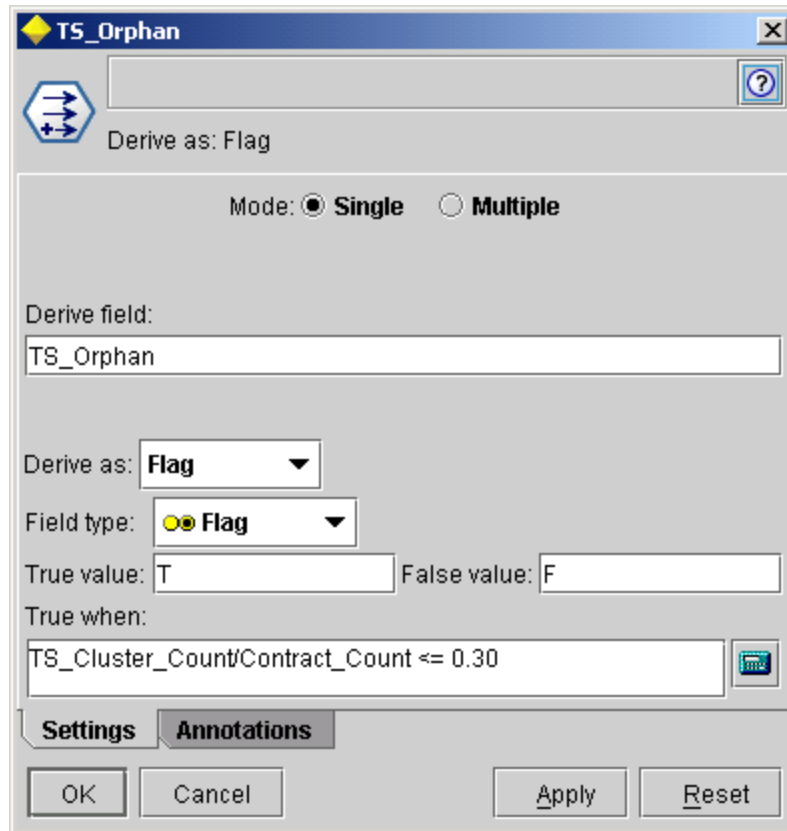


Figure 69 TS_Orphan Derive Node Settings

The final step in this stream is to select the “multiple orphans,” which is accomplished by the Triple Orphans Select Node (Figure 70). A table of these records is then produced that identifies transactions for audit.

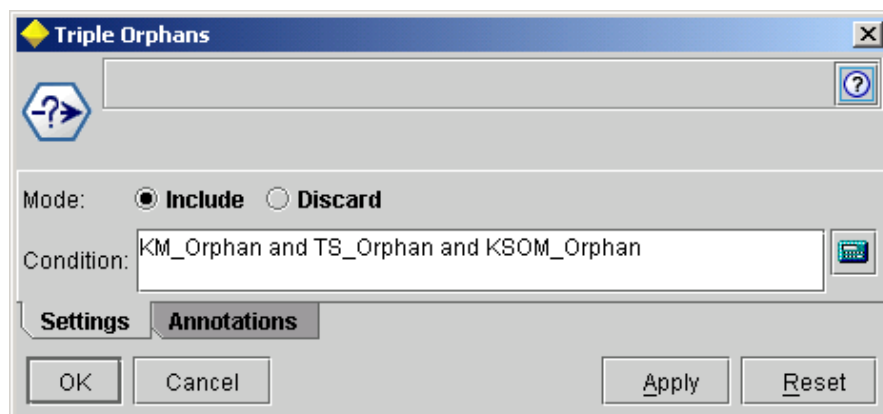


Figure 70 Triple Orphans Select Node Settings

b. Results

In addition to identifying the transactions that are “triple orphans,” analysis of the distribution of orphan transactions by cluster can give some insight into the structure of the data. The orphan distribution by cluster for the K-Means and Two Step models are shown in Figure 71 and Figure 72 respectively.

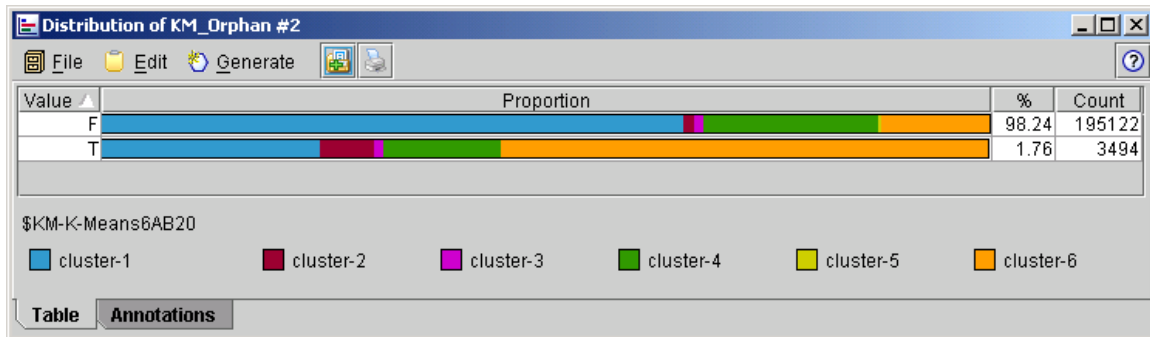


Figure 71 Distribution of Orphan Transactions by K-Means Cluster

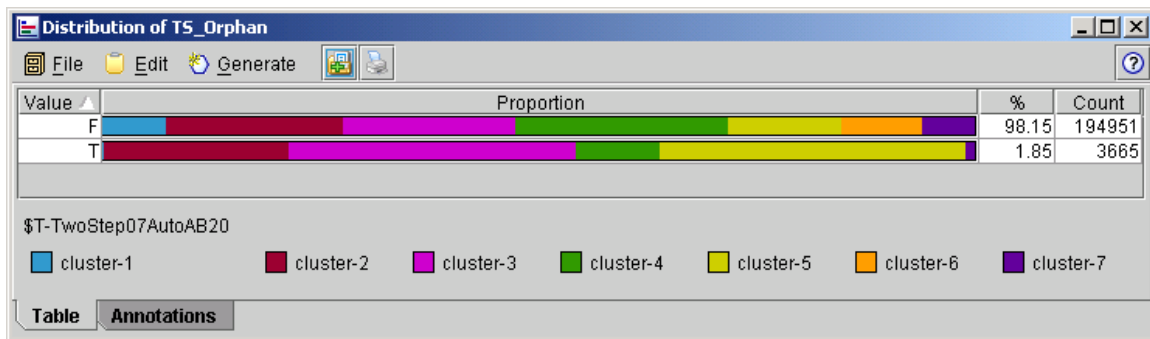


Figure 72 Distribution of Orphan Transactions by Two Step Cluster

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SPREADSHEET TOOLS FOR UNSUPERVISED MODELING

1. SUM OF SQUARES

The spreadsheet tool shown in Figure 73 is used to construct the Sum of Squares vs. Number of Clusters plot for determination of the appropriate number of clusters for K-Means modeling. It is self-explanatory and automatically produces the plot.

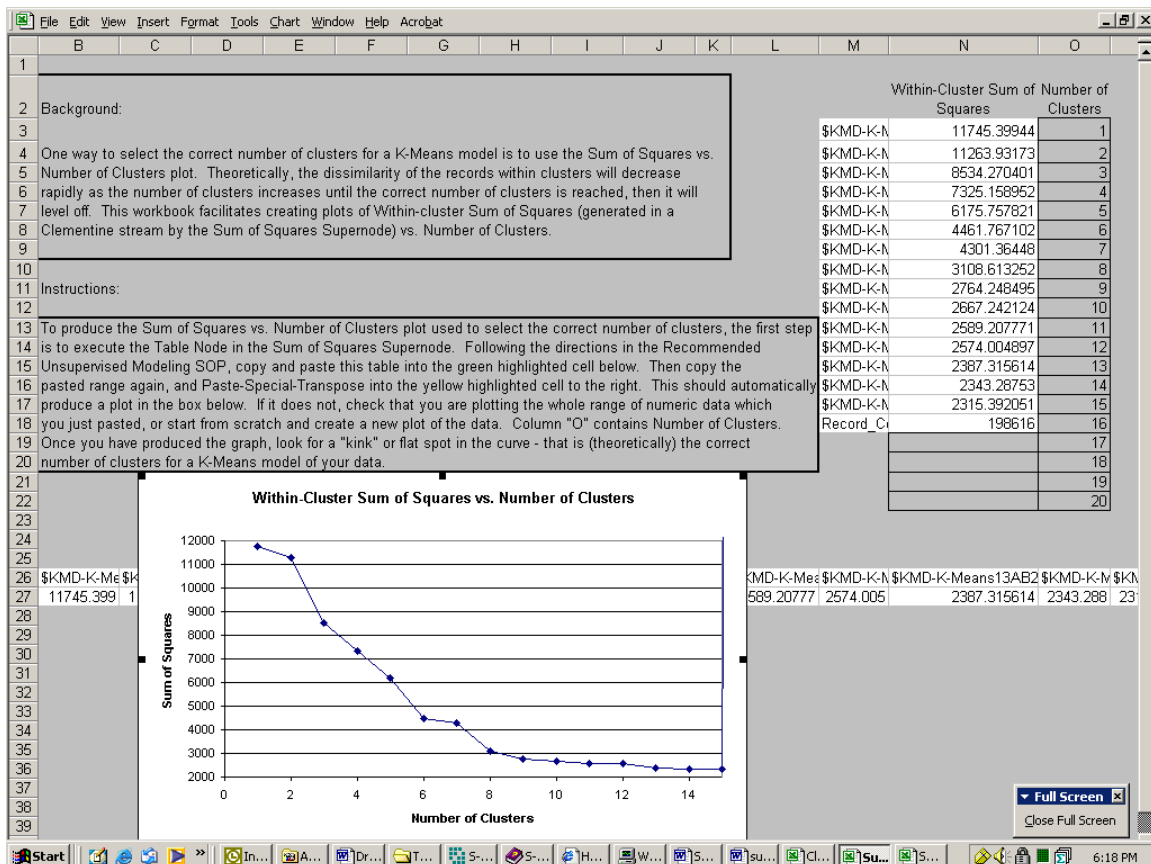


Figure 73 Sum of Squares Spreadsheet Tool

2. CLUSTER CORRESPONDENCE ANALYSIS TEMPLATE

The spreadsheet tool shown in Figure 74 and Figure 75 is used to calculate Cramer's Coefficient for selected

models. It accepts models with up to ten clusters, and automatically calculates Cramer's Coefficient and displays results for multiple models on the Analysis page. It is self-explanatory, automatic, and easy to use.

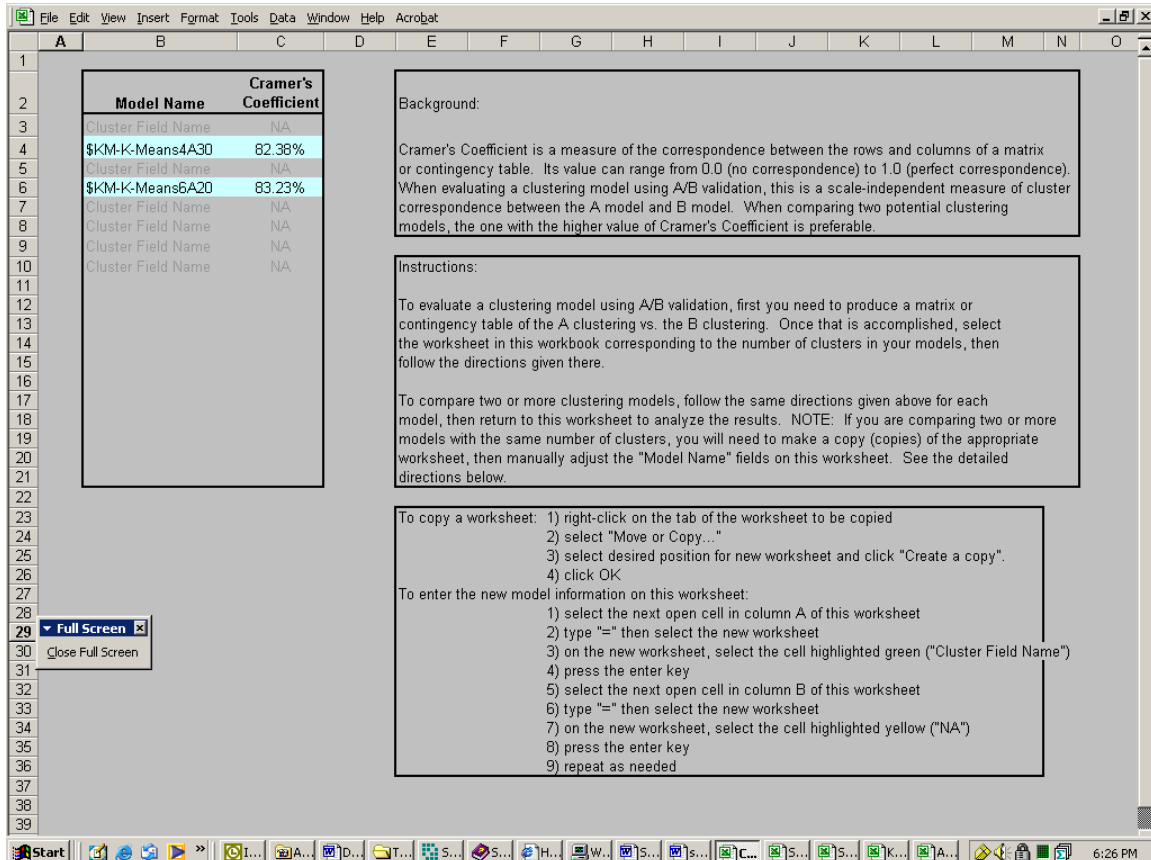


Figure 74 Cluster Correspondence Analysis Template Analysis Worksheet

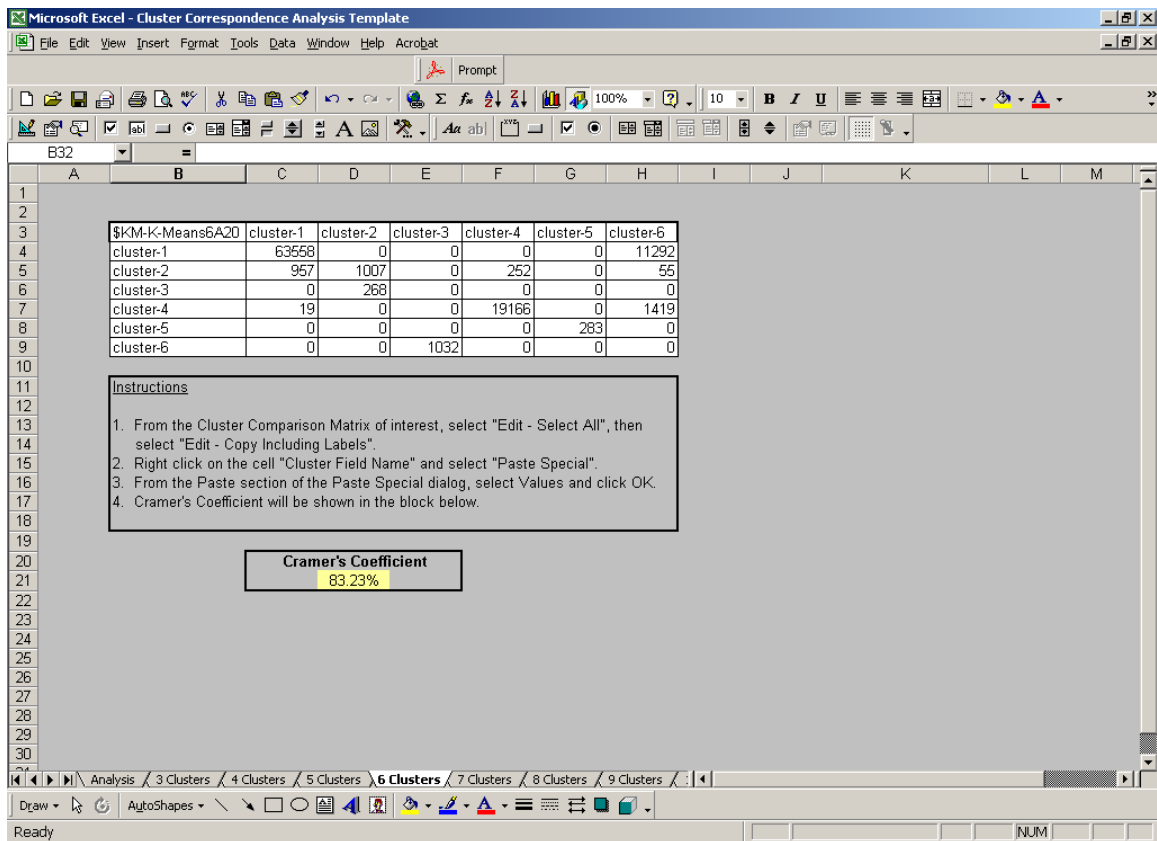


Figure 75 Cluster Correspondence Analysis Template 6 Clusters Worksheet

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. TREE CLUSTERING SPLUS IMPLEMENTATION

1. S-PLUS IRIS DATA

The S-PLUS Iris data set consists of fifty samples each of three Iris species, Setosa, Versicolor, and Virginica, with measurements of sepal length and width and petal length and width. A sample of this data is shown in Table 18. To evaluate the automatic variable selection capability of the Tree Clustering method, we add five random or "noise" variables, and multiply the values of Sepal Width and the first noise variable by ten, as shown in Table 19.

	Species	Sepal.L	Sepal.W	Petal.L	Petal.W
1	Setosa	5.1	3.5	1.4	0.2
2	Setosa	4.9	3.0	1.4	0.2
3	Setosa	4.7	3.2	1.3	0.2
...					
51	Versicolor	7.0	3.2	4.7	1.4
52	Versicolor	6.4	3.2	4.5	1.5
53	Versicolor	6.9	3.1	4.9	1.5
...					
101	Virginica	6.3	3.3	6.0	2.5
102	Virginica	5.8	2.7	5.1	1.9
103	Virginica	7.1	3.0	5.9	2.1
...					

Table 18 Example of Original Iris Data

	Species	Sepal.L	Sepal.W	Petal.L	Petal.W	N1	N2	N3	N4	N5
1	Setosa	5.1	35	1.4	0.2	64	4.8	6.8	5.5	5.4
2	Setosa	4.9	30	1.4	0.2	49	4.9	5.9	4.9	6.0
3	Setosa	4.7	32	1.3	0.2	72	4.9	4.3	5.5	6.3
...										

Table 19 Example of Scaled Iris Data With Noise Variables

2. S-PLUS IMPLEMENTATION

a. Function `tree.clust()`

The following S-PLUS function takes as input a data frame with the observations as rows and the variables as columns, and returns a list containing a list of variables retained, the size and deviance of the tree for each of those variables, and a dissimilarity matrix suitable for clustering by any S-PLUS clustering algorithm. The arguments are structured to allow flexibility in application and debugging of the function.

```
> tree.clust
function(df, fancy.dist = T, rank.y = F, verbose = F, debug = F)
{
  if(!is.data.frame(df))
    stop("This function requires a data frame")
  if(version$major < 6)
    oldClass <- class
  out <- as.data.frame(matrix(0, nrow(df), ncol(df))) # Deal with
columns whose names have embedded spaces. They suck, by the way.
#
  dimnames(out) <- dimnames(df)
  nm <- names(df)
  first.space <- first.occurrence(nm, " ")
  which <- first.space != nchar(nm)
  if(any(which)) {
    nm[which] <- substring(nm[which], 1, first.space[which] -
1)
    if(length(nm) != length(unique(nm)))
      stop("Truncating embedded spaces in names leads to
ambiguity. I give up.")
    names(df)[which] <- nm[which]
  }
#
# Handy function to convert "where" entries to leaf numbers
#
  leaf.numbers <- function(tree)
  {
    where <- tree$where
    leaves <- as.numeric(dimnames(tree$frame)[[1]])
    leaves[where]
  }
#
```

```

assign("df", df, frame = 1)
df.name <- deparse(substitute(df))
results <- matrix(0, nrow = ncol(df), ncol = 2)
dimnames(results) <- list(dimnames(df)[[2]], c("Dev", "Size"))
big.list <- vector("list", ncol(df))
if(fancy.dist) {
  big.dist.mat <- matrix(0, nrow(df), nrow(df))
}
for(i in 1:ncol(df)) {
  if(verbose > 0)
    cat("Creating tree with column", i, "\n")
  if(rank.y)
    str <- paste("tree (rank(", names(df)[i], ") ~ .,
data = df)", sep = "")
  else str <- paste("tree (", names(df)[i], " ~ ., data =
df)", sep = "")
  mytree <- eval(parse(text = str))
  if(oldClass(mytree) == "singlenode")
    next
  my.cv <- cv.tree(mytree, FUN = prune.tree)
  my.size <- my.cv$size[my.cv$dev == min(my.cv$dev)][1]
  if(my.size == 1) {
    results[i, "Dev"] <- 0
    results[i, "Size"] <- 1
    next
  }
  mytree <- prune.tree(mytree, best = my.size)
  big.list[[i]] <- mytree #
# When "fancydist" is FALSE, we simply use the leaf identifier for each
# observations. By our making it factor, daisy() will compute the
distance
# as a 0 or 1. When fancydist is TRUE, we compute the distance from
each
# observation to all the others in terms of...
#
  if(fancy.dist) {
    leaves <- leaf.numbers(mytree)
    node.numbers <-
as.numeric(dimnames(mytree$frame)[[1]])
    non.leaves <- node.numbers[!is.element(node.numbers,
leaves)]
    if(length(non.leaves) == 1) {
      dev.at.node <- mytree$frame["1", "dev"]
      names(dev.at.node) <- "1"
      subtree.dev <- deviance(mytree)
      names(subtree.dev) <- "1"
    }
    else {
      dev.at.node <- mytree$frame[mytree$frame$var !=
"<leaf>", "dev", drop = F]
      nm <- dimnames(dev.at.node)[[1]]
      dev.at.node <- as.vector(dev.at.node[, 1, drop
= T])
      names(dev.at.node) <- nm
      subtree.dev <- sapply(select.tree(mytree,
non.leaves), deviance)

```

```

    }
    u.leaves <- unique(leaves)
    lul <- length(u.leaves)
    dmat <- matrix(0, lul, lul)
    dimnames(dmat) <- list(u.leaves, u.leaves)
    for(u in 1:(lul - 1)) {
      this <- u.leaves[u]      # leaf number
      ind <- leaves == this    # logical
      for(other in (u + 1):lul) {
        that <- u.leaves[other] # leaf number
        o.ind <- leaves == that
        parent <- as.character(max(leaf.paths[this,
][match(leaf.paths[that, ], leaf.paths[this, ], 0)])) #
##cat("Distance between", this, " and ", that,
##    " is ", subtree.dev[parent],
##    ", since parent is", parent, "\n")
        dmat[u, other] <- subtree.dev[parent] #
#
# Old egad:
#           egad <- try(big.dist.mat[o.ind, ind] <-
#           big.dist.mat[o.ind, other] + subtree.dev[
#           parent]/subtree.dev["1"])
#
#           egad <- try(big.dist.mat[o.ind, ind] <-
big.dist.mat[o.ind, other] + 1 - (subtree.dev[parent]/dev.at.node[
parent]))
        if(any(is.na(big.dist.mat)))
        if(debug)
        browser()
        else stop("NA's are gonna get you")
        if(length(class(egad)) > 0 && class(egad) ==
"Error") {
          if(debug)
          browser()
          else stop("Dammit, I don't know what to do,
and debug is FALSE.")
        }
      }
    }
    dmat <- dmat + t(dmat)
  }
  out[, i] <- factor(mytree$where)
  orig.dev <- dev(df[, i])
  new.dev <- summary(mytree)$dev
  results[i, "Dev"] <- orig.dev - new.dev
  results[i, "Size"] <- my.size
  if(!is.factor(df[, i]))
    results[i, "Dev"] <- results[i, "Dev"]/var(df[, i])
}
if(!any(results[, "Size"] > 1))
  stop("Egad! No tree produced anything!")
out <- out[, results[, "Size"] > 1]
big.list <- big.list[results[, "Size"] > 1]
results <- results[results[, "Size"] > 1, ]
if(fancy.dist) {
  dists <- big.dist.mat[row(big.dist.mat) >
col(big.dist.mat)]

```

```

        attr(dists, "Size") <- nrow(df)
        attr(dists, "Labels") <- as.character(1:nrow(df))
        attr(dists, "Metric") <- "euclidean"
        oldClass(dists) <- "dissimilarity"
    }
    else dists <- daisy(out)
    out <- list(mat = out, call = match.call(), tbl = results, trees
= big.list, dists = dists)
    oldClass(out) <- "treeclust"
    return(out)
}

```

b. Application to Iris Noise Data

The following S-PLUS code was executed to produce the results shown in Chapter VI, Section D.

```

> iris.noise.scale.pam_pam(iris.noise.scale[, -1], 3, diss=F, stand=T)
> table(iris.noise[, 1], iris.noise.scale.pam$clustering)
      1  2  3
Setosa 48  2  0
Versicolor  0 17 33
Virginica  0  7 43
> iris.noise.scale.tc_tree.clust(df=iris.noise.scale[, -1])
> iris.noise.scale.tc
      Dev Size
Sepal.L 132.69178 10
Sepal.W  85.93450  4
Petal.L 144.87745  6
Petal.W 141.61504  6
Noise 1  40.08875  6
Noise 2  13.74929  2
> iris.noise.scale.tc.pam_pam(iris.noise.scale.tc$dists, 3, diss=T)
> table (iris.noise[, 1], iris.noise.scale.tc.pam$cluster)
      1  2  3
Setosa 50  0  0
Versicolor  3 47  0
Virginica  2  3 45

```

b. Application to Vendor Payment Data

The following S-PLUS code was executed to produce the results shown in Chapter VI, Section E.

```

> KB.tc_tree.clust(df=KBData[, -5])
> KB.tc
      Dev Size
CHK.AMT 321.42728  4
PMT.METH 146.32514  7
PMT.TYPE 704.36352 14
INV.AMT 372.58038  6
INV.RECV.INV.DT 410.56259  7
CHK.INV.DT 421.59578 10

```

CHK.INV.RECV.DT	415.87393	10
INTEREST	269.51651	4
DBOF	436.75056	3
OTHERX	436.47283	3
ALLX	408.59796	8
Y1.PRIOR	366.13269	4
Y1.CUR	389.61332	4
Y2.CUR.2ND	305.11099	3
Y3.PLUS	204.30980	4
ENHANCE.PAYEE	375.51852	8
STE	382.90927	6
POBOX	412.31105	15
INV.PAYEE	220.28978	8
INV.CNT	332.77792	7
DOVAMT.2K	441.00000	2
DOVAMT.1K	441.00000	2
AVG.5K	385.83321	3
PAYEE.4.PYMT	355.60706	10
INV.SEQ	67.82436	3
PMT.FREQ.HI	434.38967	8
PMT.FREQ.LO	392.99234	10
TINS	360.78780	7
MULTI.TINS	433.48543	5
MULTI.PAYEE.K	441.00000	2
MULTI.ADDR.K	441.00000	2
DISCOUNT	269.79951	4
M.PYMT	428.76615	6
MISC.OBLIG	315.41648	5
NOT.DFAR	350.46514	27
NUMADR.K	439.12694	3
NUMADREE	378.58206	14
NUM.EE.K	441.00000	3
DP109	358.23176	9
DP111	214.10555	11
MDELCKDT	379.81641	10
MDELINDT	400.30045	9
MDELIRDT	378.21893	8

```

> KB.tc.pam_pam(KB.tc$dists,4,diss=T)
> table(KBData[,5],KB.tc.pam$clustering)
      1  2  3  4
Bigsys 144 107 29 2
Opportunistic 40  3  1  1
Piggy  9  2 20  0
Smallsys 46 17 20  1

```

APPENDIX E. PROPOSED STANDARD OPERATING PROCEDURES FOR UNSUPERVISED MODELING TO DETECT FRAUD IN VENDOR PAYMENTS

1. PURPOSE AND OVERVIEW

This is a recommended Standard Operating Procedure (SOP) for Unsupervised Modeling, designed as a supplement to the Internal Review Seaside Datamining SOP. The intent of this SOP is to provide a more rigorous and standardized process for selection of unsupervised candidates in the Internal Review Datamining process. It is based on the idea that transactions that belong to the same contract are somehow similar, and thus should fall into the same cluster of a clustering model. Transactions that fall into clusters other than the one containing the majority of transactions for their contract are considered "orphans." Selection of orphan transactions is the ultimate result of this procedure.

This Recommended SOP is organized into three sections: Data Pre-Processing, Model Building and Selection, and Model Analysis. Familiarity with Clementine on the part of the reader is assumed, so some of the specific details of Clementine implementation are omitted. For more detail on any area of this SOP, refer to the Naval Postgraduate School Master's Thesis "An Improved Unsupervised Modeling Methodology for Detecting Fraud in Vendor Payment Transactions," June 2003, by Major Gregory W. Rouillard. Q:\Mongoose\Unsupervised_Modeling contains the example streams, supernodes, and spreadsheet tools referred to in this SOP.

2. DATA PRE-PROCESSING

a. Source Data and SPSS Analysis

Obtaining and opening the population database and SPSS analysis should be conducted as always, as detailed in the Datamining SOP. The procedures described in this document assume that the population database and an ODBC connection have been established, and that the Fields to Use spreadsheet has been completed.

b. The Basic Filter & Type Supernode

The Basic Filter & Type Supernode, shown in Figure 76, can be used for additional data pre-processing if desired. Note that Two Step cluster models do not admit fields with missing values, so some consideration might be given to conducting this analysis and either using filler nodes to correct missing values, or eliminating fields with a high percentage of missing values.

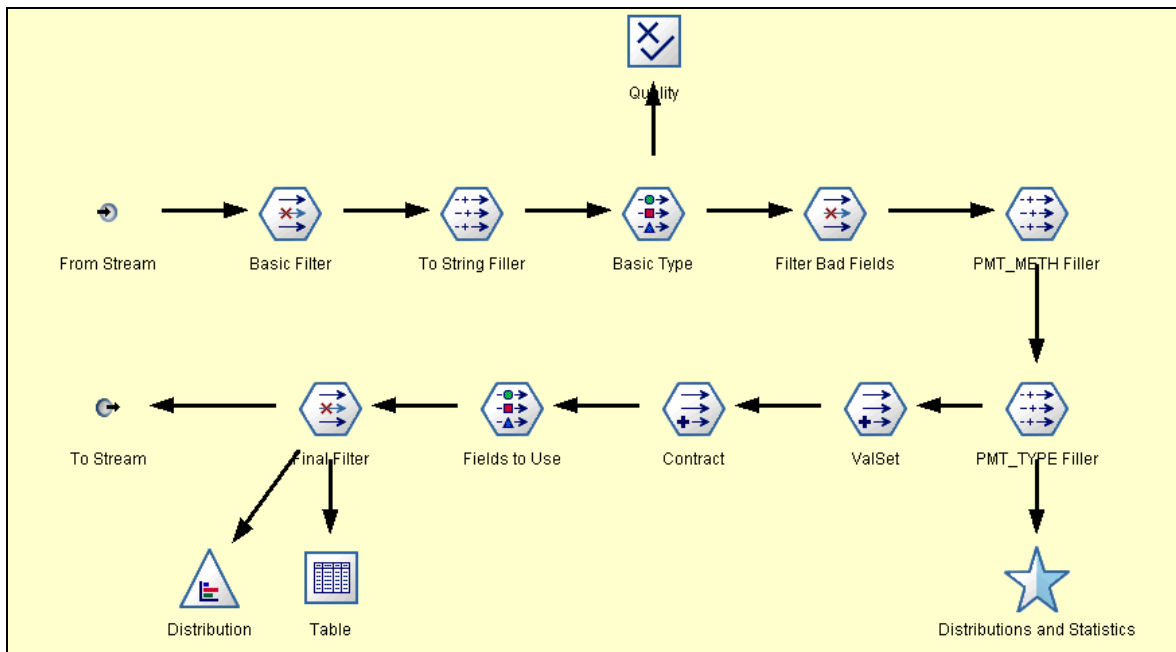


Figure 76 Basic Filter & Type Supernode

The To String Filler node is necessary to be able to browse constructed K-Means models (this is a quirk of Clementine's). The Quality node, as well as the Distributions and Statistics Supernode (Figure 77), are used to analyze fields for inclusion or filtering. The ValSet Derive Node assigns Validation Set membership of A or B depending on the value of RNDM_NUM. Its purpose is to allow the modeler to divide the data set into two equally-sized random subsets for A/B validation of K-Means and Two Step cluster models. The Contract derive node creates a new field to identify specific contracts and enable analysis of clustering results with regard to contract distributions.

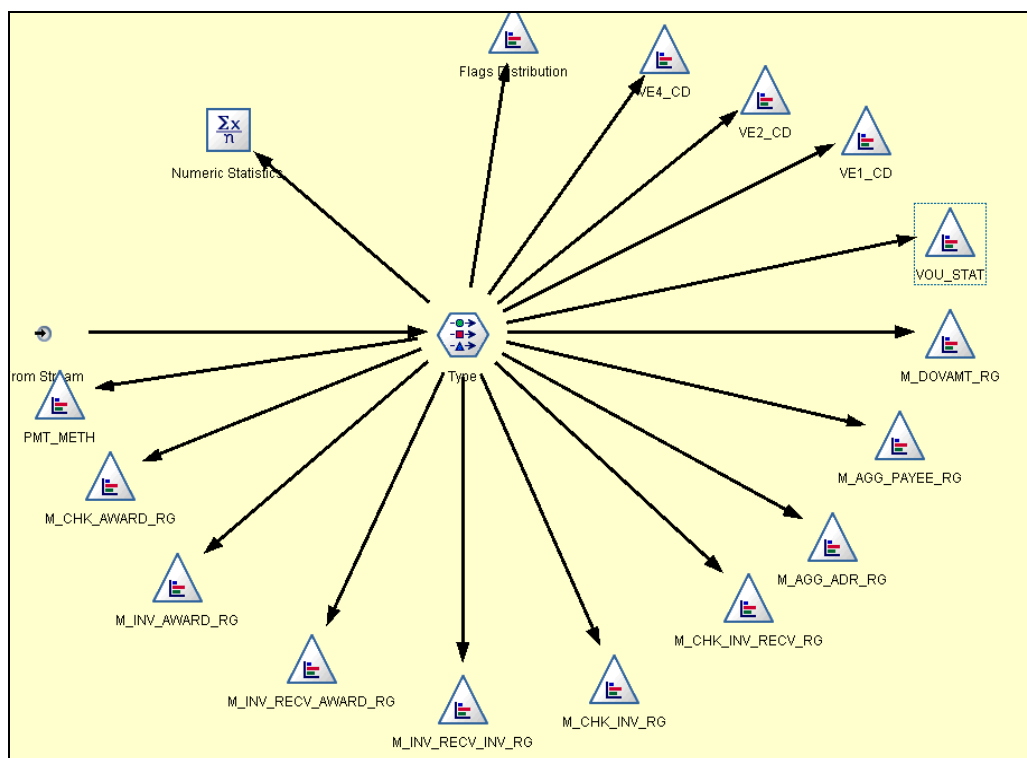


Figure 77 Distributions and Statistics Supernode

The Final Type Node is used to assign the Direction to each unfiltered field.

3. MODEL BUILDING AND SELECTION

Clementine's unsupervised modeling choices are K-Means, Two Step, and Kohonen. Refer to the Clementine 7.0 User's Guide or online Help for more details on the basic functioning and uses of these models.

a. K-Means Model Building

K-Means model building and selection is the most complicated part of the unsupervised modeling process, primarily because the modeler must select the number of clusters for model building. The procedure outlined here provides a rigorous method for selecting the correct number of clusters and validating constructed models. The stream KMeans_N02pop, shown in Figure 78, can be used as a reference for this section.

The procedures outlined here are based on theory that is fully detailed and explained in Chapter III of Major Rouillard's thesis. It is not necessary to understand this theory to successfully apply these procedures.

Note: although Clementine's K-Means modeling algorithm will accept categorical (Set and Flag Type) fields, it is not recommended. Clustering results on categorical fields tend to be arbitrary and are very sensitive to the order of the data. Always filter non-numeric fields or set their direction to "none" for K-Means clustering.

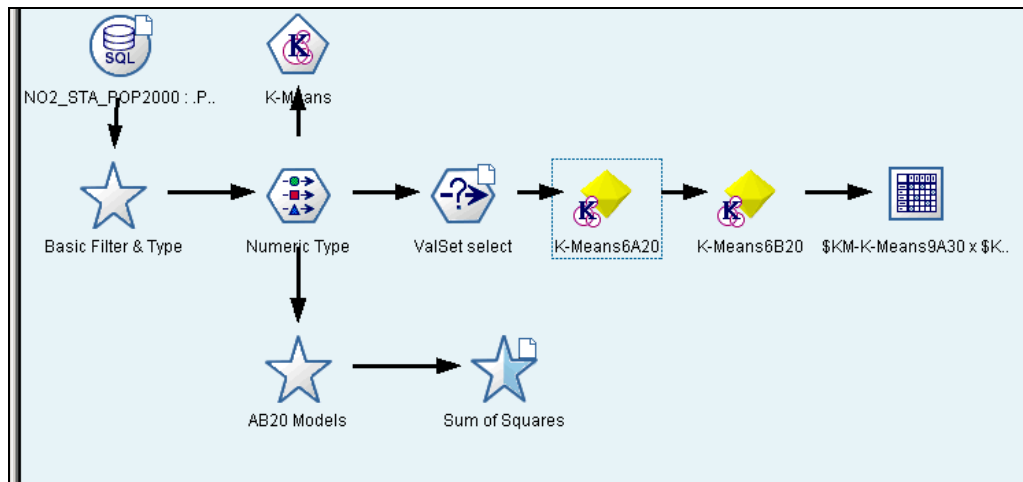


Figure 78 Kmeans_N02pop Stream

1. Build a stream with an SQL node for the source data, the Basic Filter & Type Supernode, and a Type Node setting only the numeric fields as "In." All others should be set to "None."
2. Add a K-Means modeling node downstream of the Numeric Type Node. This modeling node should only require the Simple settings, and will be used to generate all of the K-Means models.
3. Add a Select Node, shown in Figure 79, to select the appropriate Validation Set for model validation, detailed in Step 10.

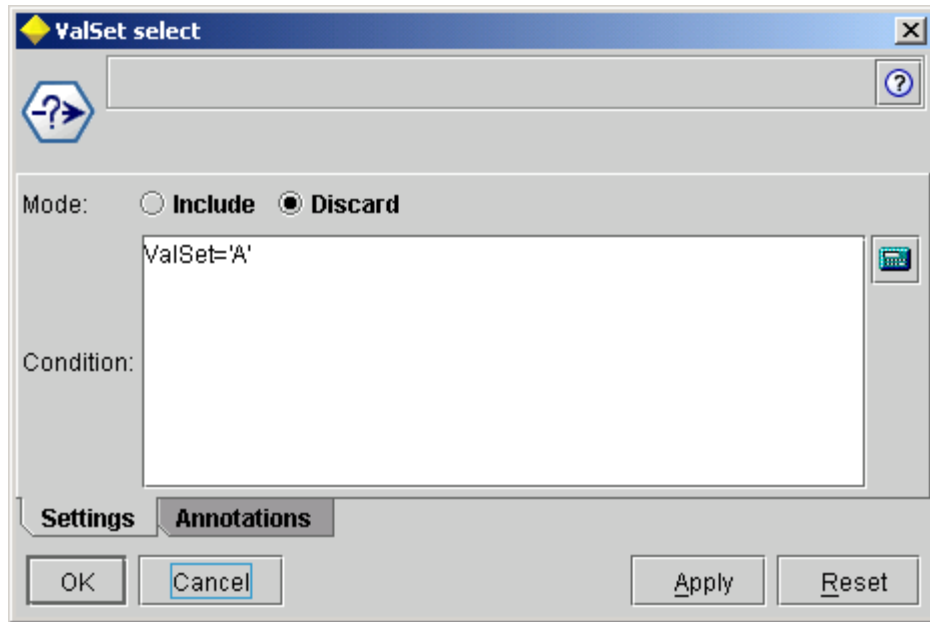


Figure 79 ValSet Select Node

4. Build models with number of clusters $k = 1, 2, 3, \dots, 15$. Stopping at $k = 10$ is usually acceptable. If you are familiar with Clementine scripting, a script such as shown in Figure 80 can be used to streamline this process. Otherwise, the models must be built by hand, changing the number of clusters and the name for each model.
5. Once all of the models have been built, connect them to the stream between the Numeric Type Node and the Sum of Squares Supernode. Figure 81 shows an example, with the models organized in a Supernode.

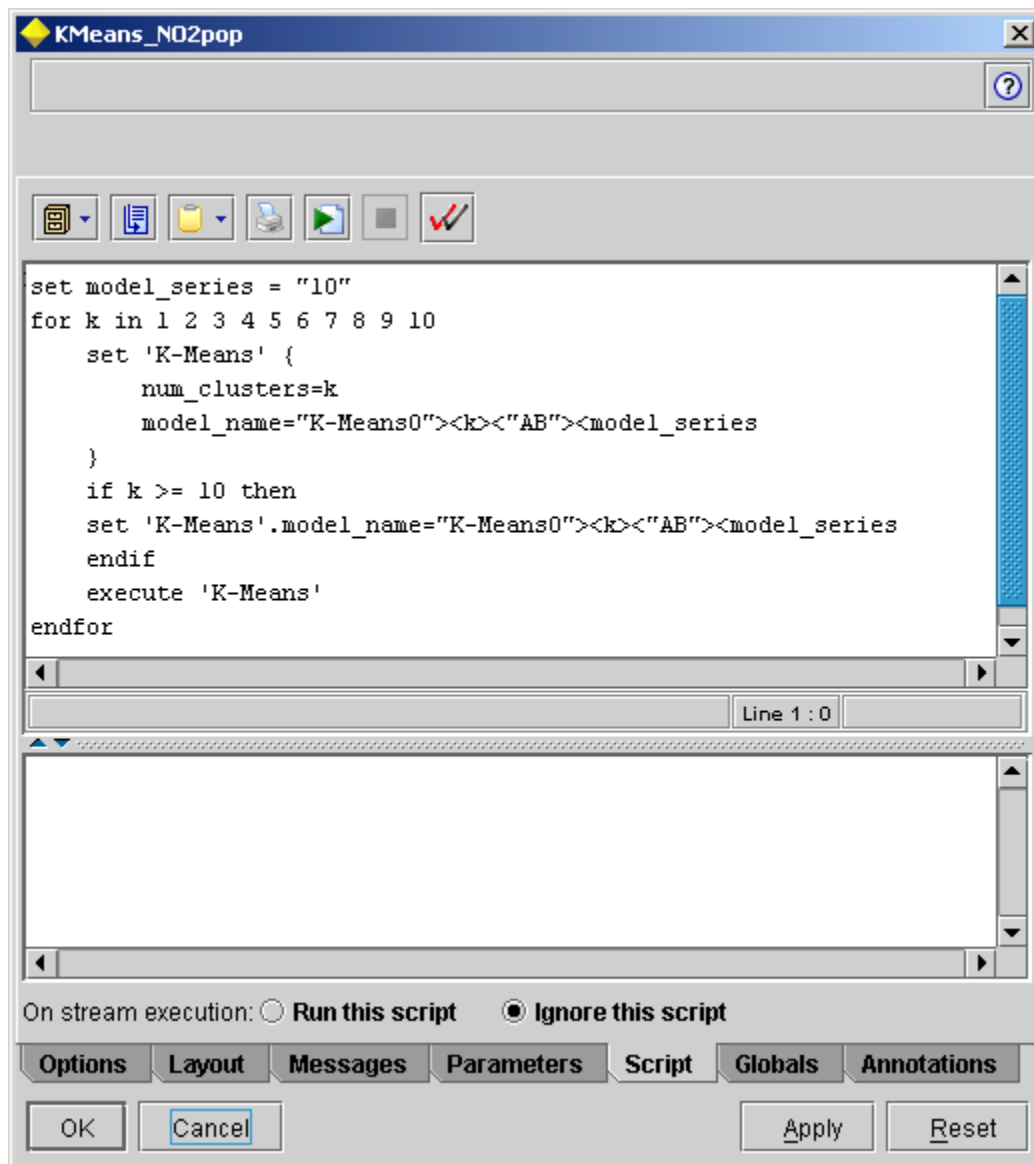


Figure 80 K-Means Model Building Script (Script Tab of the Stream Properties Dialog Box)

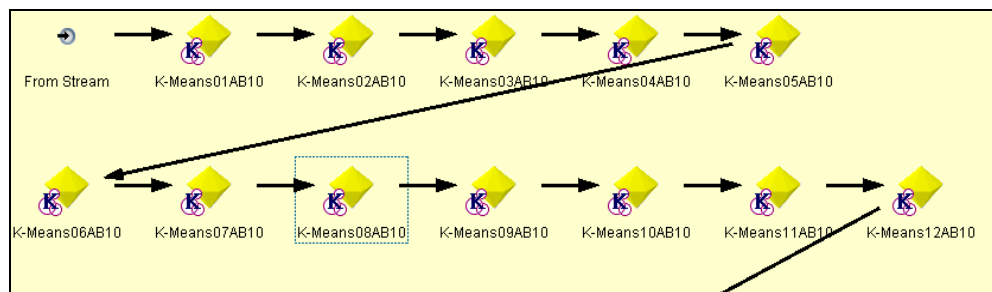


Figure 81 Generated K-Means Models

6. The Sum of Squares Supernode (Figure 82) produces a table of values that is used in the spreadsheet Sum Of Squares to help determine the correct number of clusters for K-Means modeling.

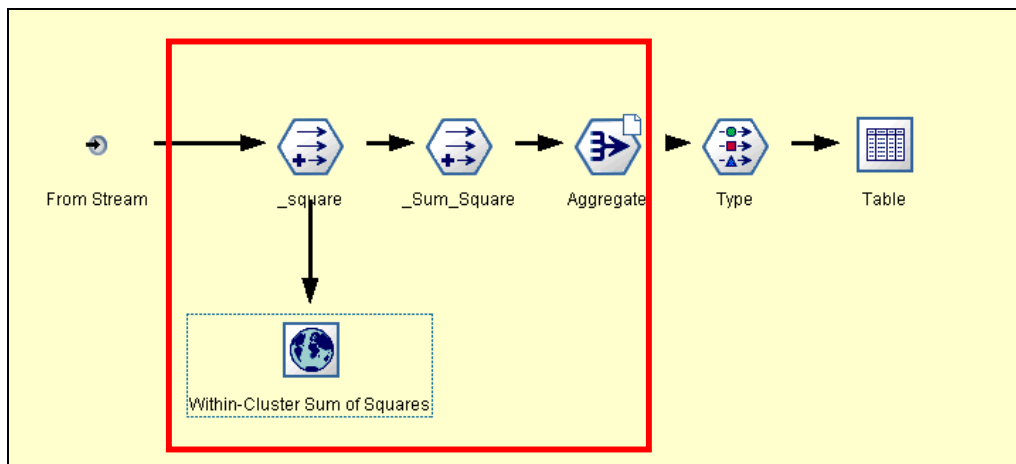


Figure 82 Sum of Squares Supernode

7. The four nodes boxed in the above figure must be edited to select the correct fields. Figure 83, Figure 84, Figure 85, and Figure 86 show examples of this step.

_square

Derive as: Formula

Mode: ☐ Single ☒ Multiple

Derive from:

- \$KMD-K-Means01AB10
- \$KMD-K-Means02AB10
- \$KMD-K-Means03AB10

Field name extension: Add as: ☒ Suffix ☐ Prefix

Derive as: **Formula** TIP: Refer to selected fields by using @FIELD

Field type: **Range**

Formula:

$(to_number(@FIELD))^{**2}$

Settings **Annotations**

OK Cancel Apply Reset

Figure 83 _Square Derive Node Dialog Box

Within-Cluster Sum of Squares

Globals to be created:

Field	MEAN	SUM	MIN	MAX	SDEV
\$KMD-K-Means01AB10_square	☐	☒	☐	☐	☐
\$KMD-K-Means02AB10_square	☐	☒	☐	☐	☐
\$KMD-K-Means03AB10_square	☐	☒	☐	☐	☐
\$KMD-K-Means04AB10_square	☐	☒	☐	☐	☐
\$KMD-K-Means05AB10_square	☐	☒	☐	☐	☐
\$KMD-K-Means06AB10_square	☐	☒	☐	☐	☐

Default operation(s): ☐ MEAN ☒ SUM ☐ MIN ☐ MAX ☐ SDEV

☒ Clear all globals before executing

☐ Display preview of globals created after execution

Settings **Annotations**

OK **Execute** Cancel Apply Reset

Figure 84 Within-Cluster Sum of Squares Set Globals Dialog Box

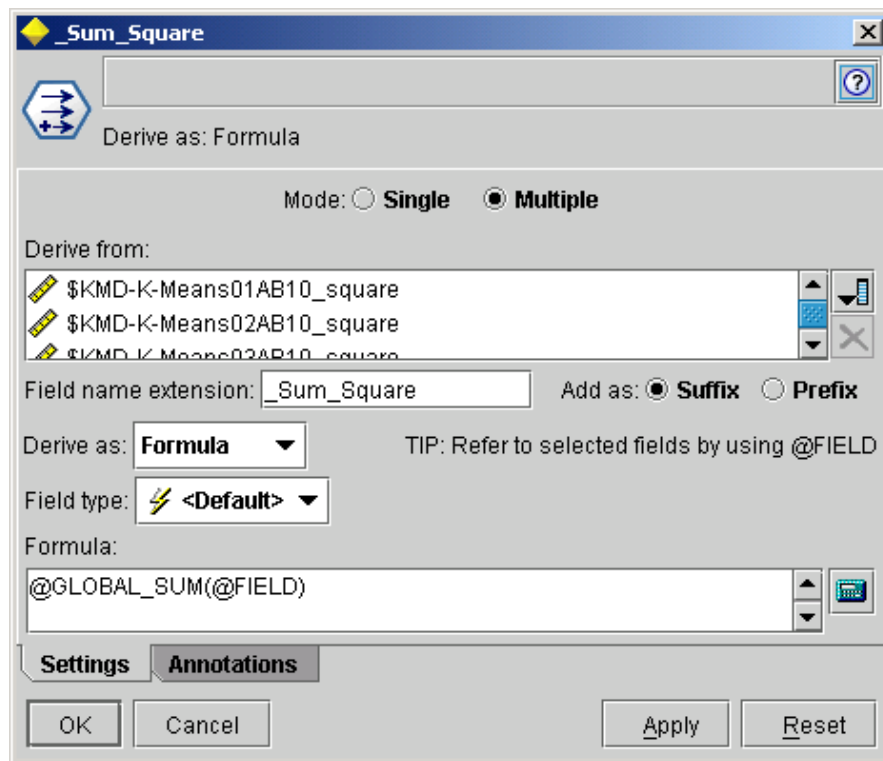


Figure 85 _Sum_Square Derive Node Dialog Box

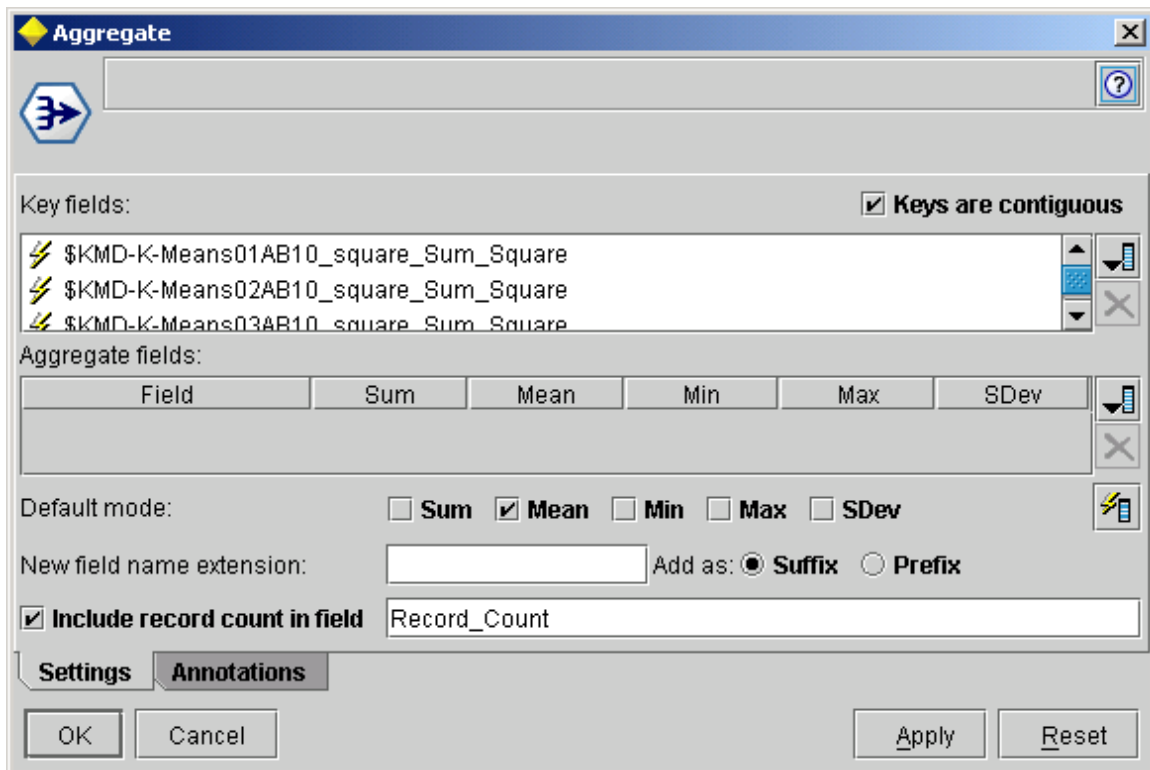


Figure 86 Sum of Squares Aggregate Node Dialog Box

8. When the Table Node is executed, it produces a table containing a field for each model that is its Within-Cluster Sum of Squares. In this table, select Edit-Select All, then Edit-Copy (inc. field names). Open the Sum of Squares spreadsheet, and then follow the directions given therein. The end result is a graph similar to the one shown in Figure 87, and the correct number of clusters is at the “kink” or flat spot in the curve.

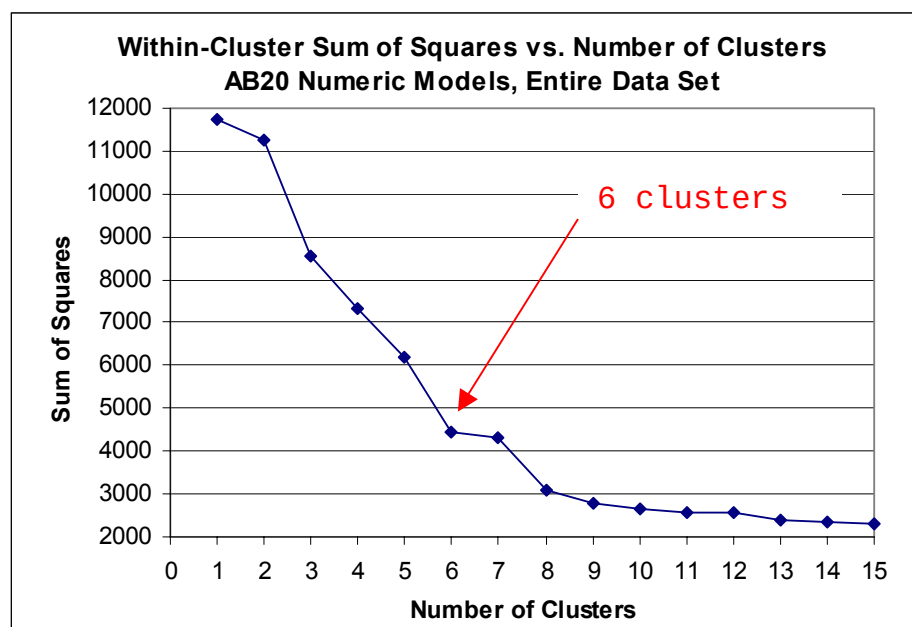


Figure 87 Example of Sum of Squares Plot

9. After selecting the correct number of clusters for K-Means modeling, the next step is to validate that model for comparison with other generated models.
10. Connect the K-Means Model Node to the ValSet Select Node, and build a model on the “A” Validation Set with the number of clusters selected in Step 8. Next, build a model with the same number of clusters

on the “B” Validation Set. Finally, connect both models to the ValSet Node (still set to select Validation Set “B”) and connect a Matrix to compare the cluster fields \$KM-<model name>, as shown in Figure 88.

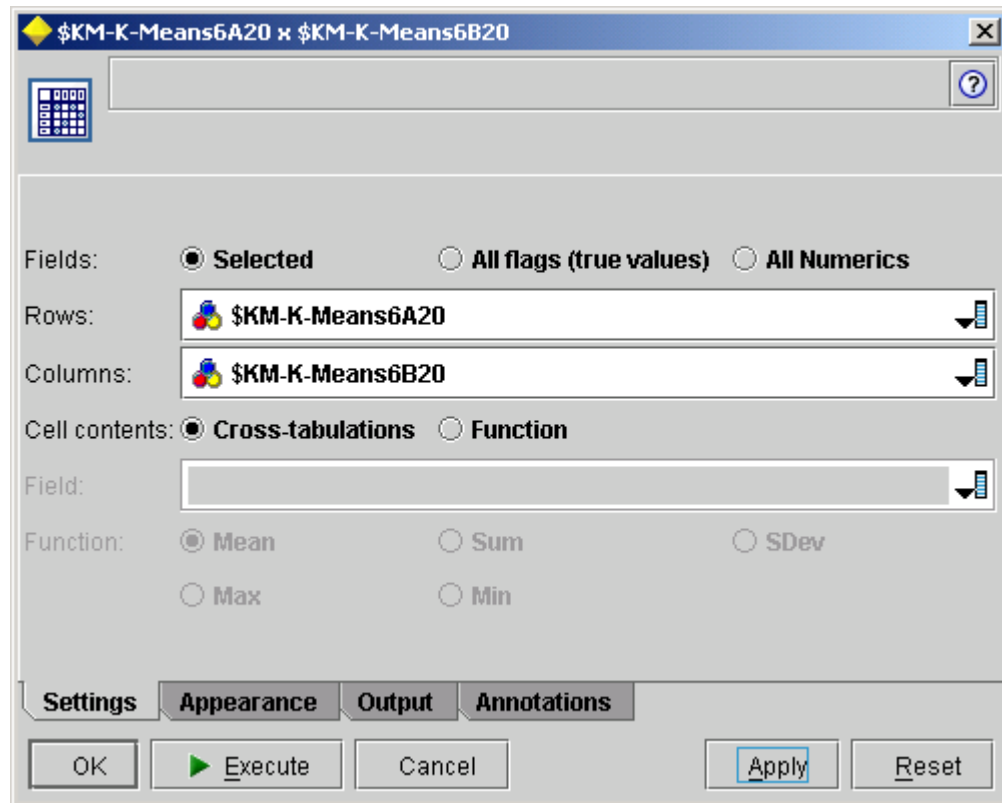


Figure 88 Matrix Node Settings Tab

11. This matrix will show the cross-tabulation (co-clustering) of the two models. Select Edit-Select All and Edit-Copy (inc. field names), then paste into the appropriate workbook in the spreadsheet Cluster Correspondence Analysis Template. Be sure to save your workbook under another name to keep the template clear. Follow the instructions in that

spreadsheet, and you will have a single number (Cramer's Coefficient, a measure of how good your clustering model is) to compare with other models.

b. Two Step Model Building

Building a Two Step model is much easier than building the "right" K-Means model. Two Step models are designed to work with all data types, so the Two Step modeling node may be connected directly to the Basic Filter & Type Node. Build a Two Step model with the simple (default) settings, then rename it to incorporate the number of clusters (automatically chosen by Two Step). The validation procedure is the same as described in Steps 9, 10, and 11 of the preceding section.

c. Kohonen Model Building

Building a Kohonen model is not difficult, but it can be very time- and memory- intensive, and there are many expert options which can affect the results. It is recommended to use the default settings of the Expert Model Tab, changing only the dimensions of the generated map (Figure 89).

Trial and error may be required to determine appropriate dimensions for the Kohonen map. Generally speaking, for a data set the size of the audit populations, a map of size 10x10 or larger should be considered for interpretability.

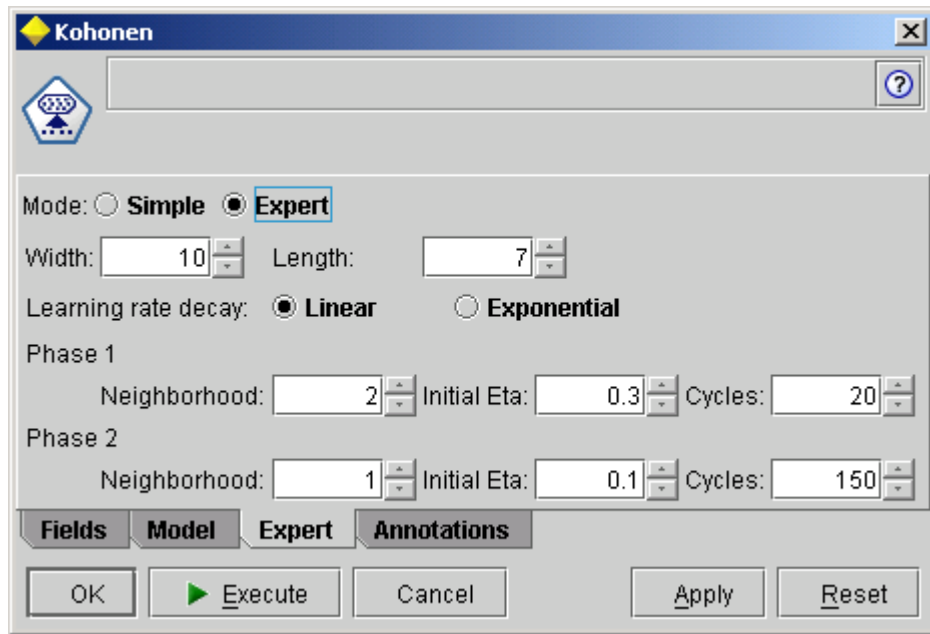


Figure 89 Kohonen Model Node Expert Tab

4. MODEL ANALYSIS AND RESULTS

The stream `Model_analysis_N02pop` (Figure 90) demonstrates the method described in this section. In this stream, each generated model selects its own orphan transactions, and the transactions that are selected by all three models are forwarded for audit.

The selection of orphans is highly dependent on determination of the threshold for contract concentration. For example, a model that selects as orphans only transactions falling in clusters containing 30% or fewer of the transactions in a contract will identify more orphans than one whose threshold is 10%. This threshold is set in the Derive Nodes of the Orphans Supernode (Figure 94), discussed in detail below.

The concept of orphan transactions in a Kohonen mapping is not as simple as for a K-Means or Two Step

cluster model, because by the nature of a Kohonen map there is not necessarily a “home” node for each contract. Therefore it might be desirable to evaluate a Kohonen map based on the concept of “sparse” nodes (ones with few records): perhaps the transactions that occupy sparse nodes are more interesting than those in dense nodes. The Sparse Prototypes Supernode facilitates this type of analysis.

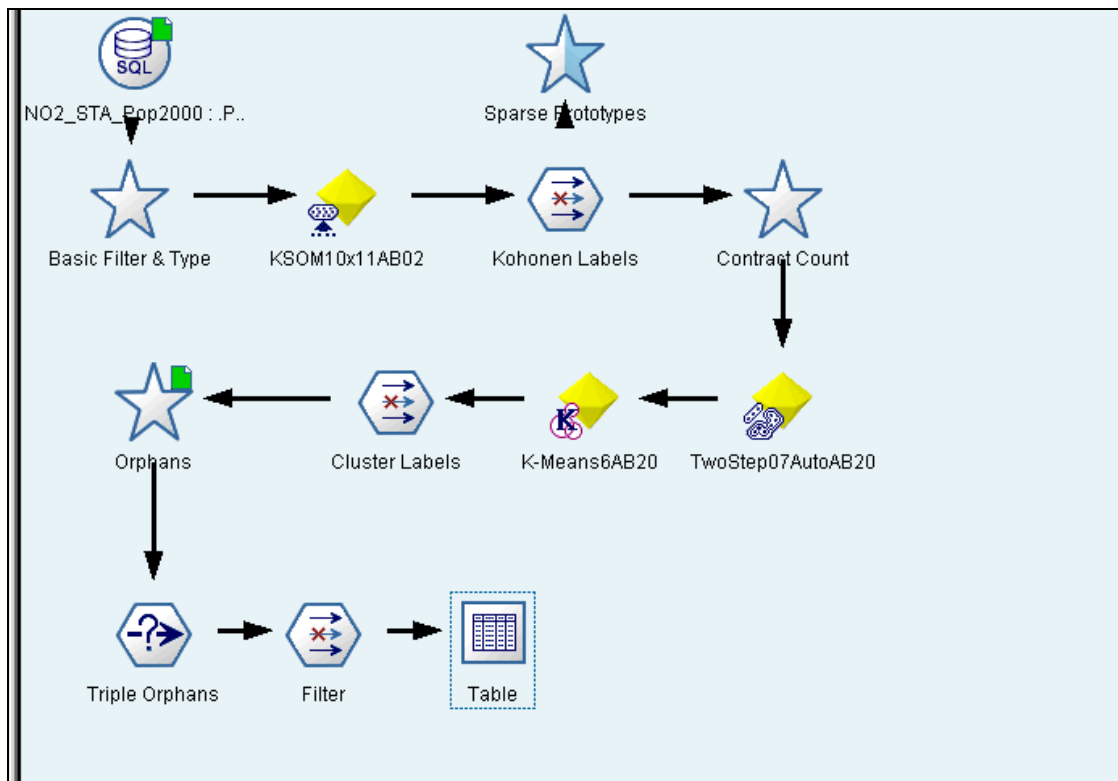


Figure 90 Model_analysis_N02pop

The Contract Count Supernode (Figure 91) produces a field containing the number of transactions in the contract to which each record belongs, which is essential to identifying orphan transactions.

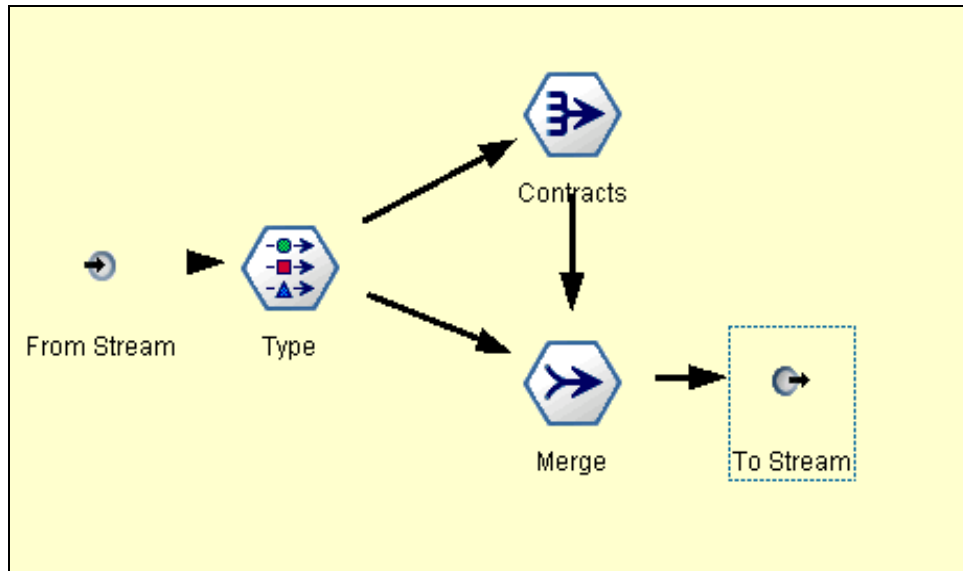


Figure 91 Contract Count Supernode

The data is first aggregated by contract, and then the Merge Node (Figure 92 and Figure 93) creates a new field with the number of contracts for each transaction.

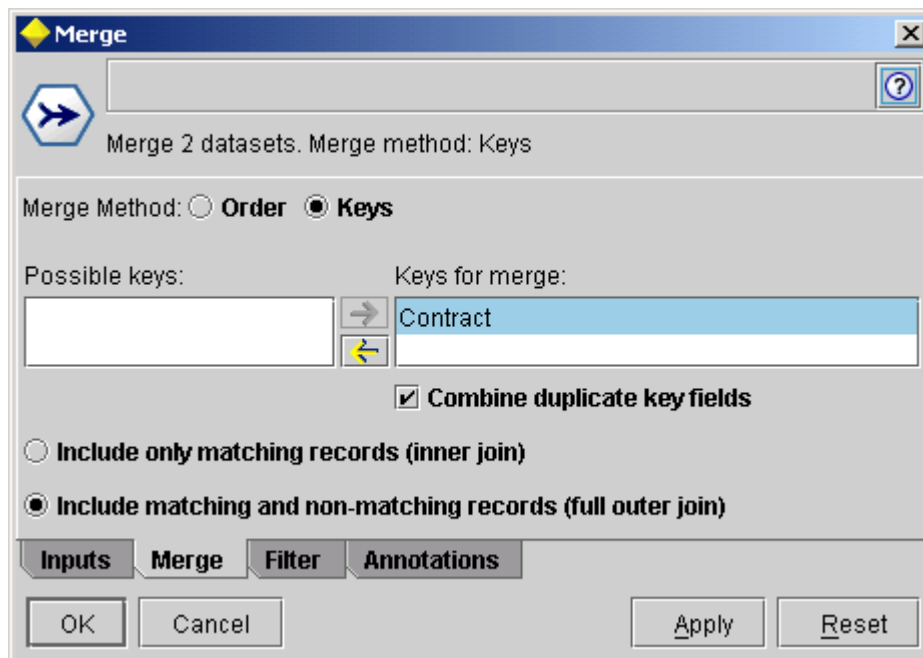


Figure 92 Merge Node Dialog Box, Merge Tab

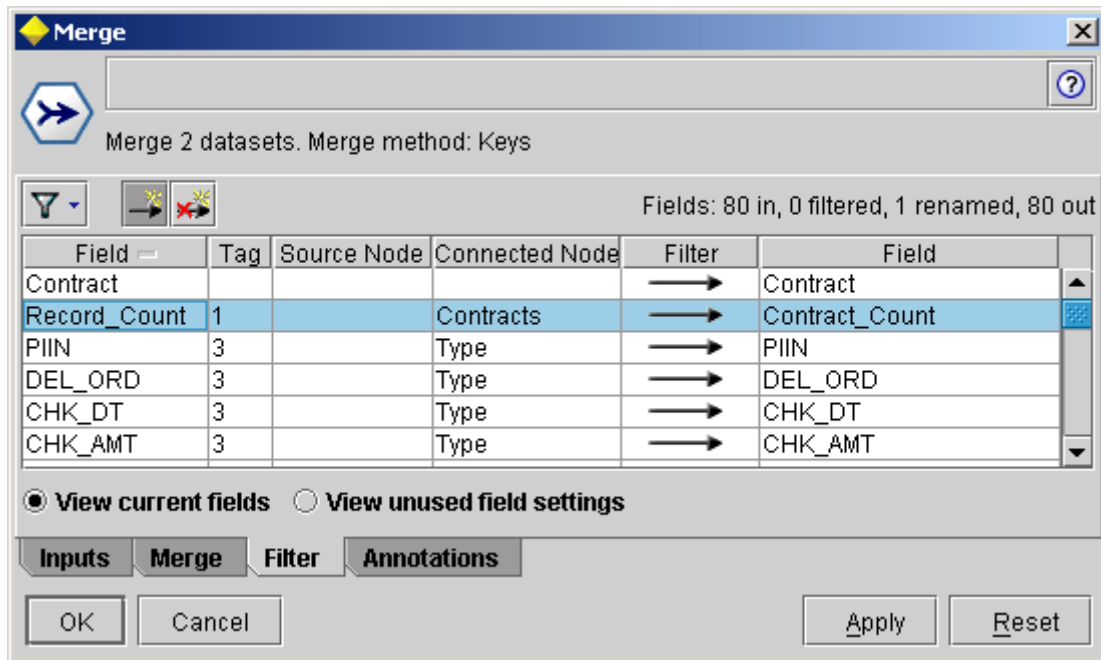


Figure 93 Merge Node Dialog Box, Filter Tab

The Orphans Supernode (Figure 94) accomplishes the important task of creating fields identifying records as orphans for one or more of the generated models. For each type of model, Two Step, K-Means, and Kohonen, the data is first merged on contract and cluster number (prototype number in the Kohonen case), then merged back to create a field identifying the number of transactions in each cluster from each contract. Figure 95 shows an example Merge Node Filter Tab, with the new field TS_Cluster_Count. The other two merge nodes are very similar and produce the new fields KM_Cluster_Count and KSOM_Prototype_Count.

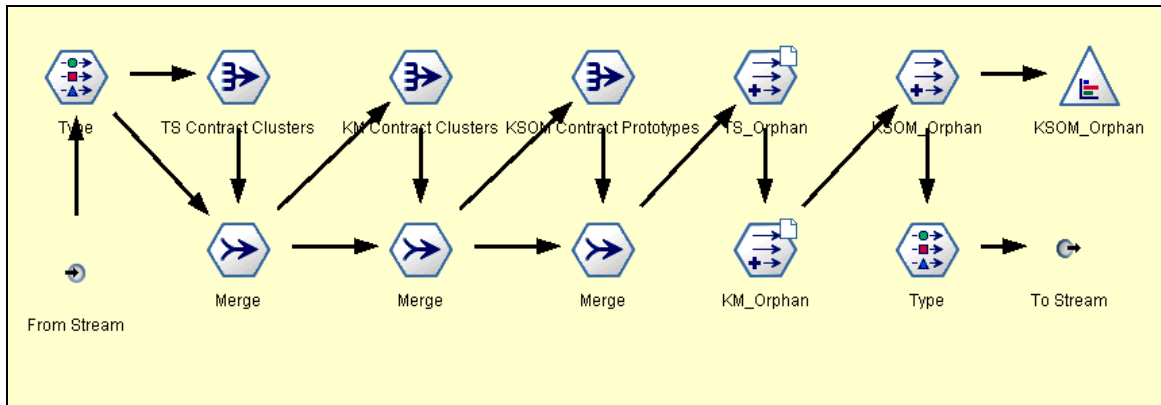


Figure 94 Orphans Supernode

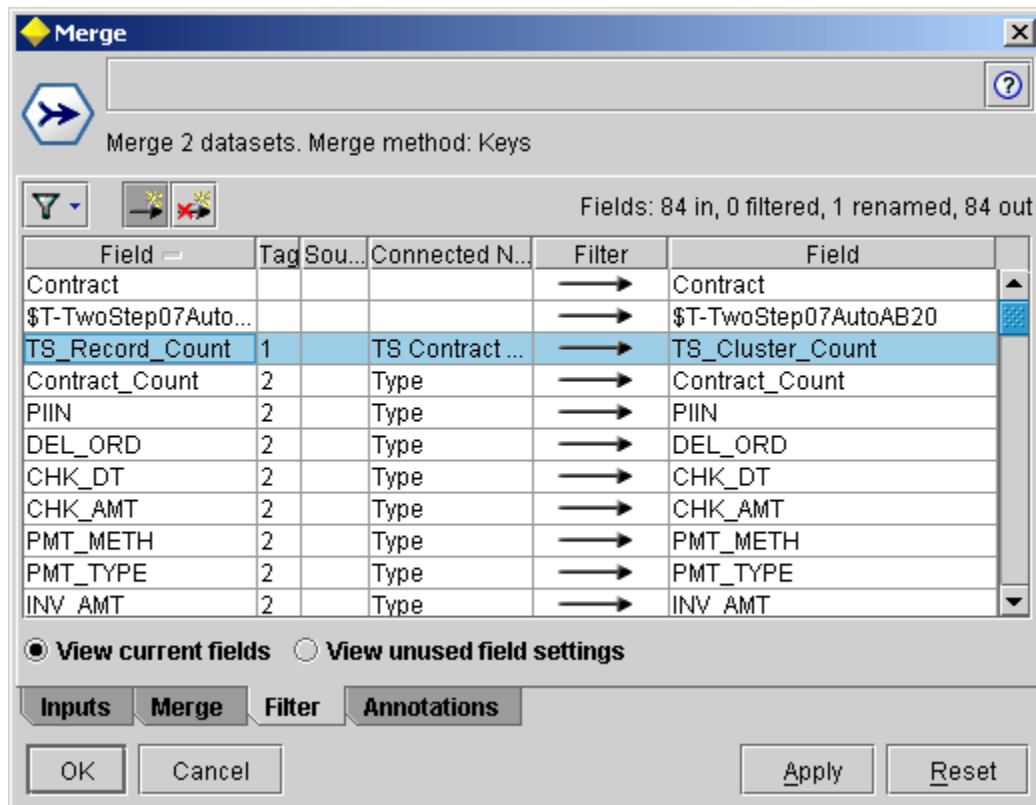


Figure 95 Merge Node Filter Settings

The three Derive Nodes create new Flag fields to identify orphan transactions. Figure 96 shows an example for the Two Step orphans; the other two derive nodes are very similar. As discussed at the beginning of this

section, selection of the orphan threshold (30% shown here) has a large impact on the number of orphans identified.

The screenshot shows a dialog box titled "TS_Orphan". At the top left is a yellow diamond icon with a blue arrow. Below it is a blue hexagonal icon with three white arrows pointing right. To the right of these icons is a text field containing "Derive as: Flag" and a question mark icon. Below this is a section with two radio buttons: "Single" (selected) and "Multiple". Underneath is the label "Derive field:" followed by a text field containing "TS_Orphan". Below that is the label "Derive as:" followed by a dropdown menu showing "Flag". Next is the label "Field type:" followed by a dropdown menu showing "Flag" with a yellow circle icon. Below these are two text fields: "True value:" containing "T" and "False value:" containing "F". Underneath is the label "True when:" followed by a text field containing the expression "TS_Cluster_Count/Contract_Count <= 0.30" and a small icon. At the bottom are two tabs: "Settings" (selected) and "Annotations". Below the tabs are four buttons: "OK", "Cancel", "Apply", and "Reset".

Figure 96 TS_Orphan Derive Node Settings

The final step in this stream is to select the "multiple orphans," which is accomplished by the Triple Orphans Select Node (Figure 97). A table of these records is then produced that identifies transactions for audit.

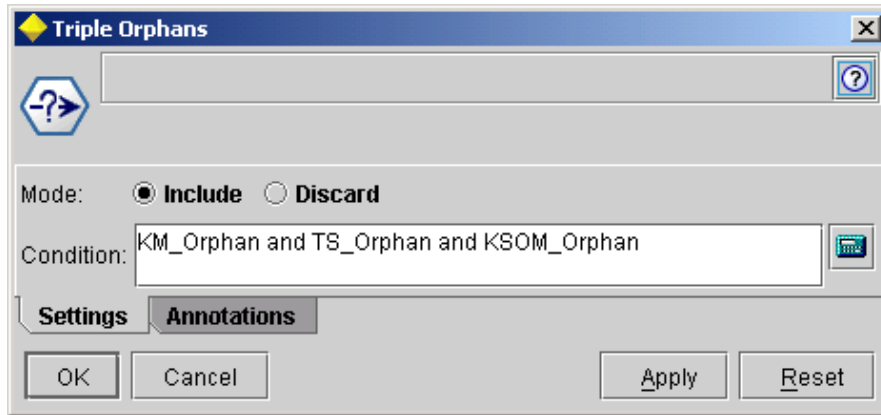


Figure 97 Triple Orphans Select Node Settings

The final step is to generate a table of the selected transactions. Alternatively, a Derive or Select Node can be generated from this table to select these transactions based on SEQ_ID or some other field.

LIST OF REFERENCES

- [1] Breiman, L., and others, *Classification and Regression Trees*, pp. 59-63, Wadsworth International Group, 1984.
- [2] Buttrey, Samuel E., "A Scale-Independent Clustering Method With Automatic Variable Selection Based On Trees," draft Naval Postgraduate School Technical Report, 2003.
- [3] Chiu, Tom Y.M., and others, "A Two-step Clustering Algorithm for Mining Large Datasets with Mixed Type Attributes," *Proceedings of AFCEA International's 18th Annual Federal Database Colloquium and Exposition*, pp. 55-67, 2001.
- [4] Clementine Datamining Software, Version 7.0, SPSS, Inc., 2002.
- [5] Conover, W.J., *Practical Nonparametric Statistics, Third Edition*, pp. 229-230, John Wiley & Sons, Inc., 1999.
- [6] Defense Finance and Accounting Service, Task Order 0033, *Improper Payments/Data Mining Project Support Final Report*, 2000.
- [7] Defense Finance and Accounting Service, *Standard Operating Procedures for Datamining*, 2003.
- [8] Gordon, A.D., *Classification, Second Edition*, pp. 183-204, Chapman & Hall/CRC, 1999.
- [9] Gower, J.C., "A General Coefficient of Similarity and Some of its Properties," *Biometrics*, Vol. 27, pp. 857-871, 1971.

- [10] Hastie, T., Tibshirani, R., and Friedman, J., *Elements of Statistical Learning*, pp. 453-485, Springer, 2001.
- [11] Holmes, Susan, "Classification Trees," [<http://www-stat.stanford.edu/~susan/courses/b494/index/node45.html>], January 2002.
- [12] Kauffman, L., and Rousseeuw, P., *Finding Groups in Data*, pp. 1-44, John Wiley & Sons, Inc., 1990.
- [13] Monteiro, Antonio, *Multiple Additive Regression Trees, A Methodology For Predictive Data Mining For Fraud Detection*, Master's Thesis, Operations Research Department, Naval Postgraduate School, September 2002.
- [14] S-PLUS 2000 Statistical and Data Analysis Software, Insightful Corporation, 2000.
- [15] Zaiane, O.R., and others, "On Data Clustering Analysis: Scalability, Constraints, and Validation," [<http://www.cs.ualberta.ca/~zaiane/postscript/pakddZaiane.pdf>], 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, VA
2. Dudley Knox Library
Naval Postgraduate School
Monterey, CA
3. Marine Corps Representative
Naval Postgraduate School
Monterey, California
4. Director, Training and Education
MCCDC, Code C46
Quantico, Virginia
5. Director, Marine Corps Research Center
MCCDC, Code C40RC
Quantico, Virginia
6. Marine Corps Tactical Systems Support Activity
(Attn: Operations Officer)
Camp Pendleton, California
7. Director, Studies and Analysis Division
MCCDC, Code C45
Quantico, Virginia
8. Internal Review Seaside (Operation Mongoose)
DOD Center Monterey Bay
Seaside, California