



New Developments for CART® With Insurance Examples

Presented at the 2005 Casualty Actuarial Society Predictive Modeling
Special Interest Seminar, Chicago, IL September, 2005

Dan Steinberg,



CART: Classification and Regression Trees

- 1984 Monograph by BFOS:
 - Leo Breiman (1928-2005) Probability theorist, National Academy of Sciences
 - Jerome Friedman, Physicist, numerical methods, National Academy of Sciences
 - Richard Olshen, Mathematical Statistics, Bioinformatics
 - Charles Stone, Probability theorist, National Academy of Sciences
- Arguably still the best exposition of decision tree philosophy, mathematical foundations, use in practice
- Reports on research conducted since mid-1970s
- CART name trademarked by its authors in 1987 in conjunction with commercialization of their proprietary software
 - Only one true CART (and true MARS® also trademarked)
 - Actual code is a trade secret and written by Jerome Friedman one of the world's most brilliant code writers

CART: Brief 20 year perspective

- CART described in 1984, first commercial software released in 1987
 - A few early adopters in statistics, banking, DoD, computer science, engineering
 - Mostly very slow realization of the power of the technology
 - By 1990 though over 1,000 scholarly articles referencing CART
- General strengths now understood to be:
 - Ability to develop models rapidly and largely automatically
 - Strong in face of dirty and incomplete data (even 95% missing in every column)
 - Reveals problems in data visually (superb data error detection)
 - Reveals interactions readily
 - Performance generally on par with logistic regression and GLM
 - Sometimes noticeably better, sometimes worse, no definite pattern
 - Some problems for which CART is decidedly the tool of choice

CART vs GLM

- When the data is “dirty”
- When there are many missing values
- When interactions (context dependent effects) are important
- Researchers in health insurance report

“In medical data everything interacts with everything else”

- May have relevance to other areas also

CART: some further benefits and features

- Trees can be grown to reflect costs of misclassification:
 - Is a false negative worse than a false positive
 - Eg, in identifying higher risk applicants a false negative insures a high risk while a false positive rejects the business
 - CART will grow an entirely different tree to reflect asymmetric costs
- Linear combination (oblique) splits
 - Splitters are weighted averages of predictors and can reflect complex logic in a compact way
 - LCs can also capture linear structure

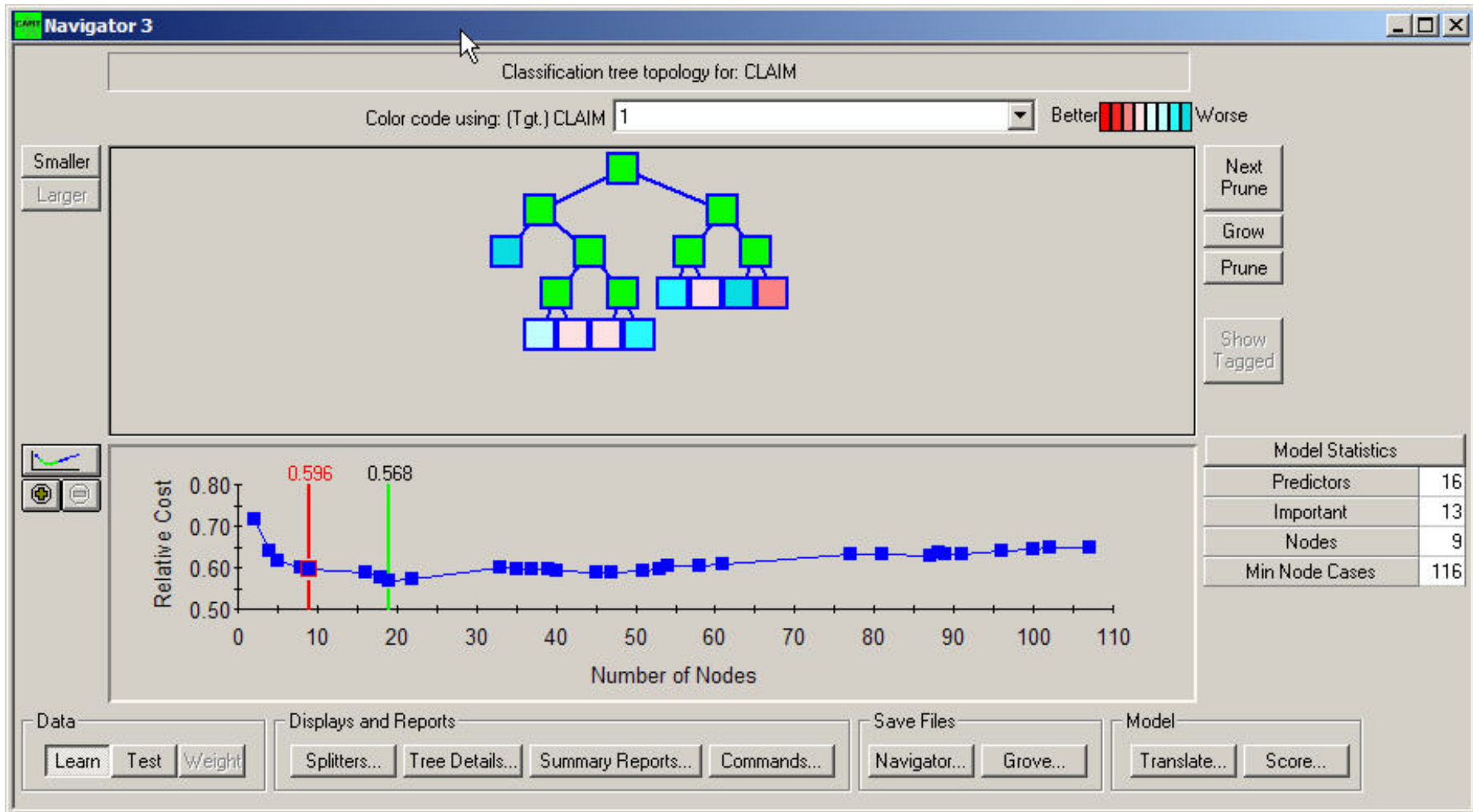
CART: Evolution of the technology

- Improved methods to assess tree performance node-by-node
- Improved controls over tree generation
 - Forcing predictors into specific nodes
 - Penalizing “high cost” predictors
 - Structuring trees by controlling the order in which predictors appear
- Multiple tree methods:
 - Using hundreds to thousands of trees to capture data structure
 - Offer by far the best accuracy seen across a broad range of problems
 - Can capture any data structure no matter how complex or simple (linear)

CART: An Insurance Example

- Models required to predict “Probability of Claim”
- Specific Sample of Policy Holders
- Core predictors:
 - Demographics (age, sex, geographic location)
 - Acquisition channel (how customer acquired)
 - Product characteristics (coverage and options selected, number insured)
- Data organization:
 - Data available for several years
 - Use data on policies issued in Period 1 to predict “ANYCLAIM” in a later year
 - Some policy holders not observed for entire 12 months

Example of Basic CART Model: Any Claim

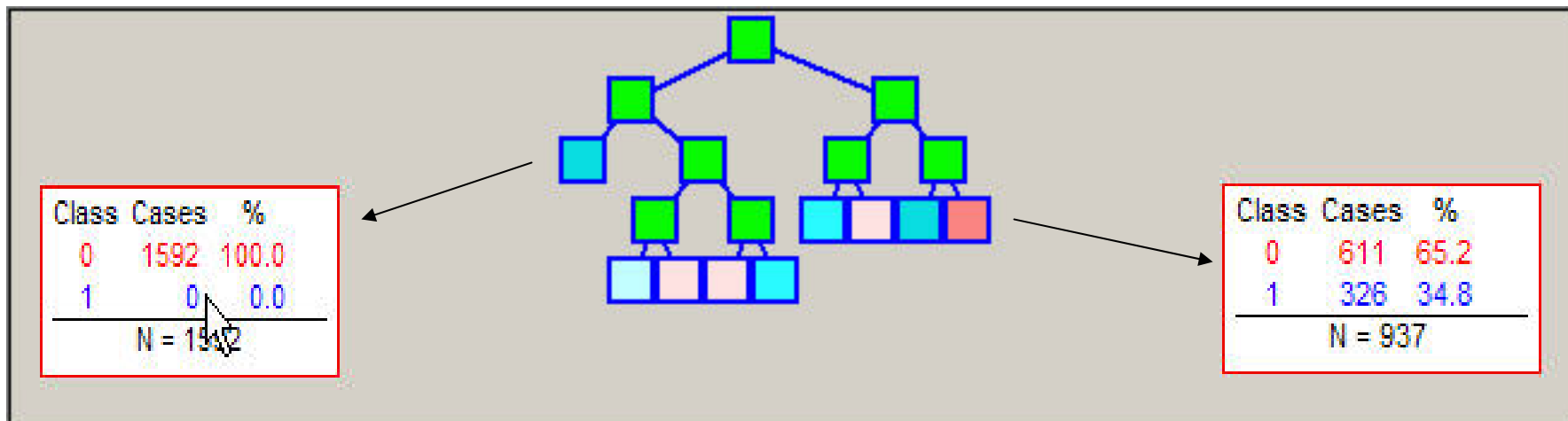


Top portion displays tree topology and bottom displays performance as a function of tree size

CART: Tree Topology and TEST data results

Sample: 7.2% claims
11,349 cases total

Class	Cases	%
0	10535	92.8
1	814	7.2
<hr/>		
N = 11349		



Red nodes indicate high concentration of target class

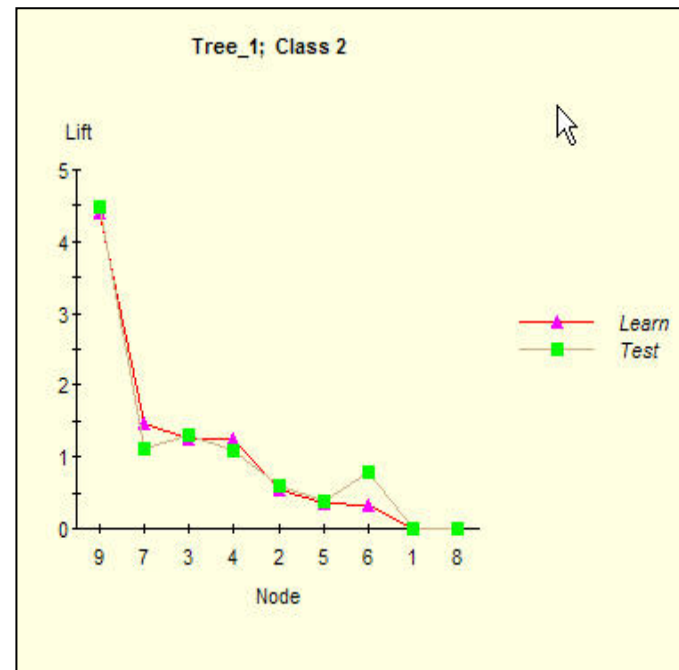
Blue nodes indicate low concentration

Red node at right bottom expanded in box at right

Note how well tree separates non-claimants at bottom left and identifies high claim risk segment on bottom right

CART: Train/Test Comparisons of Lift (Relative Risk)

Node	Node Type	Direction Agreement	Learn Lift	Test Lift	Class Learn count	Class Test count	Other Learn Count
Tree_1	9 nodes	Agree					
1	Terminal	Agree	0.00	0.01	0.00	1.00	1,592.00
2	Terminal	Agree	0.54	0.59	26.00	27.00	589.00
3	Terminal	Agree	1.25	1.30	346.00	328.00	3,165.00
4	Terminal	Agree	1.24	1.09	29.00	24.00	268.00
5	Terminal	Agree	0.34	0.38	94.00	99.00	3,381.00
6	Terminal	Agree	0.33	0.79	3.00	7.00	113.00
7	Terminal	Agree	1.47	1.12	63.00	41.00	481.00
8	Terminal	Agree	0.00	0.00	0.00	0.00	142.00
9	Terminal	Agree	4.40	4.48	326.00	287.00	611.00

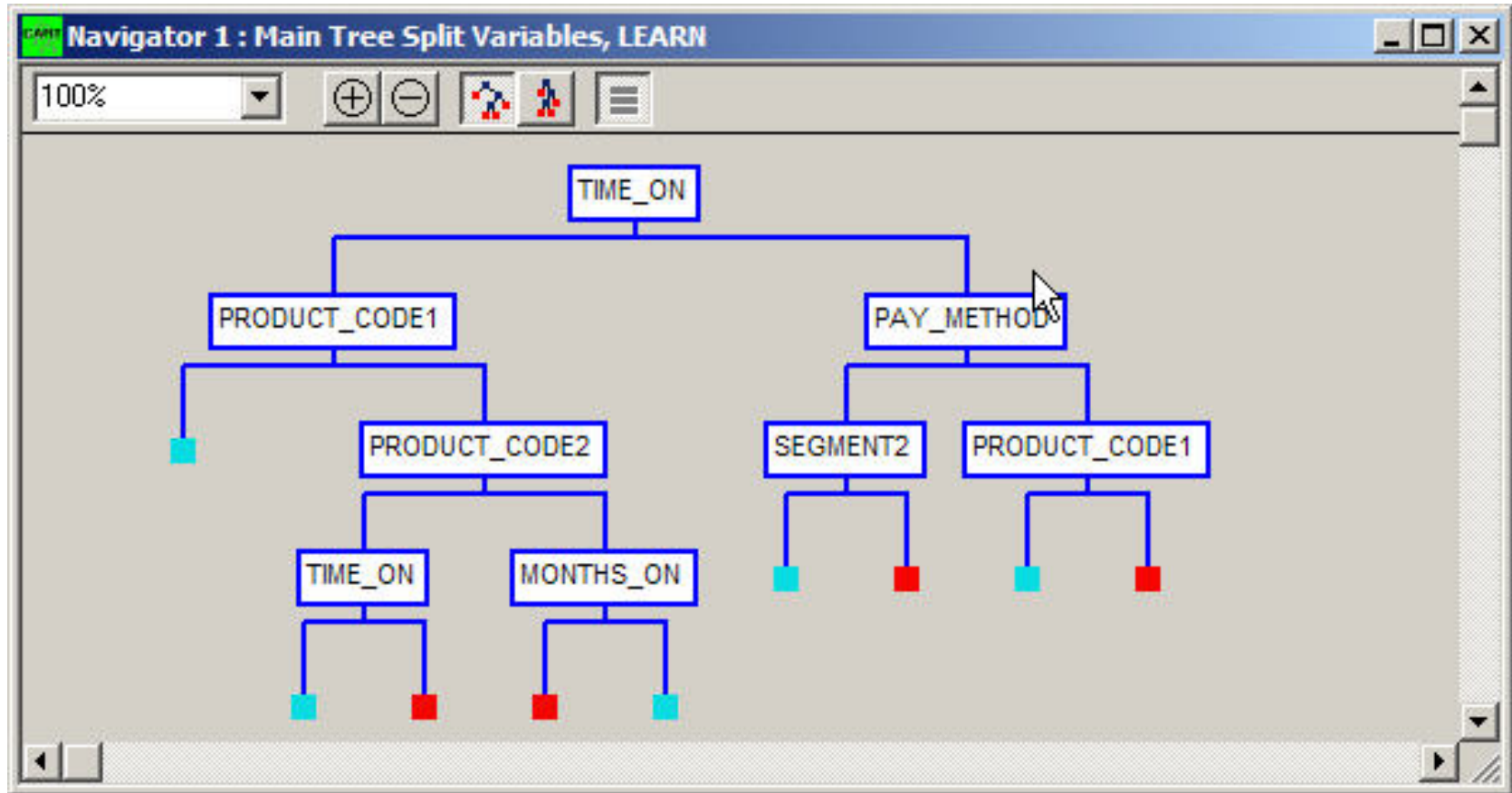


For every tree we produce a train/test comparison chart listing train and test lift (relative risk) in every node.

The contents of the lift columns are displayed in the graph on the right.

Note all but one node have very close train/test agreement on lift.

CART: Original Tree



Tree begins with an exposure to risk measure and ends with Pay_Method and Product characteristics

Controlling where predictors may appear

Splitter Variable Constraint Criteria

Depth In Tree

Number Of Cases
Min Cases: 0
Max Cases: 0
0 = no constraint

Primary Splitter
Surrogate Splitter

Sort: File Order

25 Min Depth Max 0

Specified Variables will not be permitted in the RED region

Constrained/StructuredSM CART Tree

- Regulate where and in what order predictors can appear in a tree (patent pending methodology)
- Most often used to limit one set of predictors in top of tree and require that the bottom of the tree switch to a different set of predictors
- We define 3 regions in the tree which are displayed on the next tree

Splitter Variable Disallow Criteria

Variable	1	2	3	Ind.	Min Cases	Max Cases
AGE_YRS	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
CLAIM	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
COST_FACTOR	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
GENDER	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
MONTHS_ON	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
N_COVERED	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
PAY_METHOD	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
PAY_PERIOD\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
POST_CODE_BIN	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
PRICE_FACTOR	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
PRODUCT_CODE1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
PRODUCT_CODE2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0

Sort:

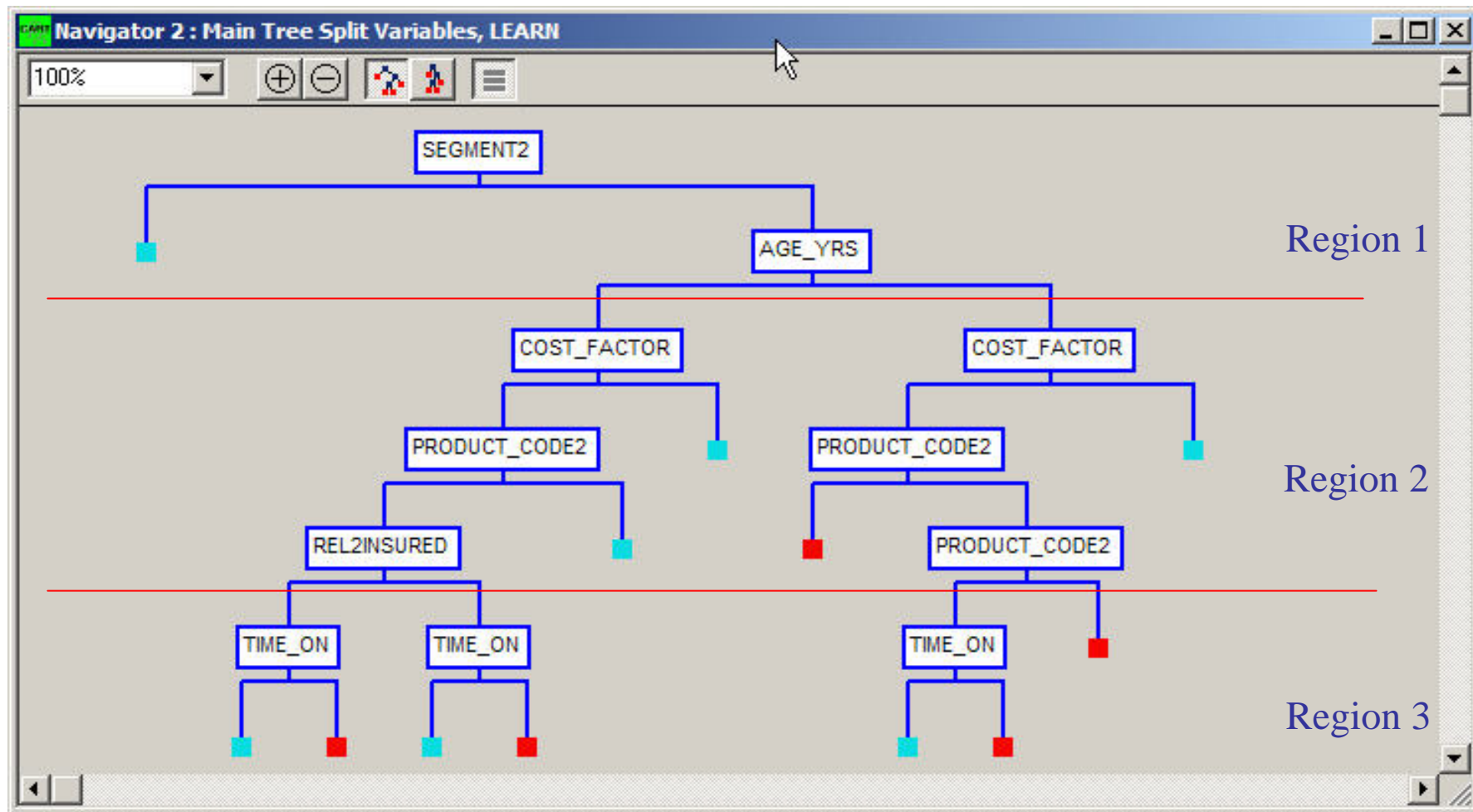
Primary Splitter
Surrogate Splitter

Disallow Split Region

1	2	3	Ind.
Split Disallowed Above Depth			
<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>
<input type="text" value="3"/>	<input type="text" value="3"/>	<input type="text" value="0"/>	<input type="text" value="0"/>
Split Disallowed Below Depth			
<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="Ind."/>

Constrained/Structured CART Tree: Example

- We have 3 regions: Top for demographics, center for policy related attributes and bottom for exposure measure



Constrained Vs Unconstrained Performance

Unconstrained

Actual Class	Total Cases	Percent Correct	0 N=6122	1 N=5227
0	10,535	56.84	5,988	4,547
1	814	83.54	134	680
Total:	11349			
	Avg.:	70.19		
	Overall % Correct:	58.75		

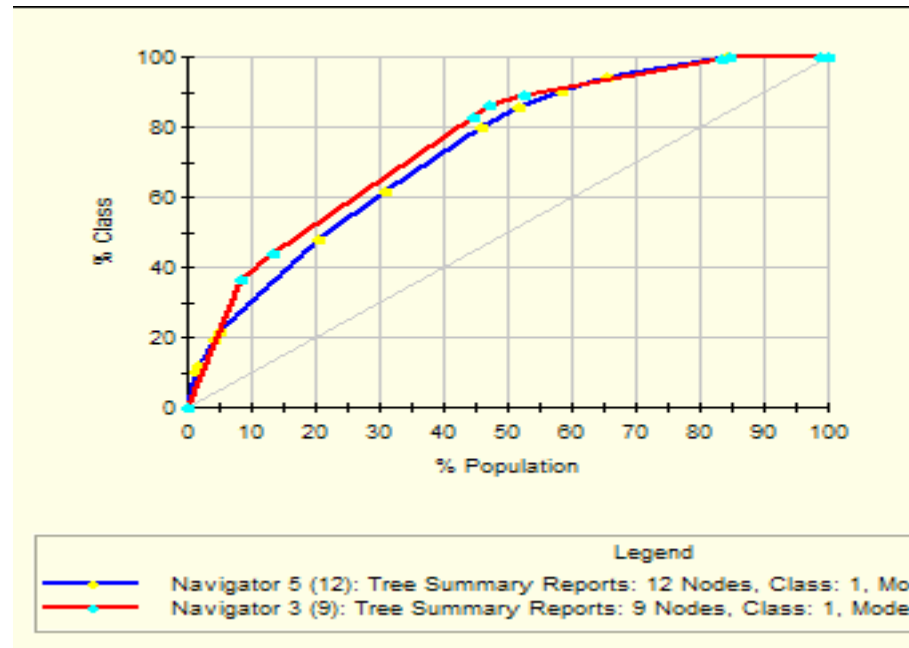


Constrained

Actual Class	Total Cases	Percent Correct	0 N=5487	1 N=5862
0	10,535	50.97	5,370	5,165
1	814	85.63	117	697
Total:	11349			
	Avg.:	68.30		
	Overall % Correct:	53.46		



Constrained and Unconstrained Trees



Another comparison of the two trees

ROC unconstrained: .782 ROC constrained: .752

Multiple Trees: Ensembles or Committees of Experts

- Multi-tree methods were first mentioned in the literature in the early 1990s
- Breiman introduced the “Bagger” bootstrap aggregation in 1995
- Freund and Schapire introduced “boosting” in 1996
- Friedman introduced MART (Multiple Additive Regression Trees) in 1999
 - Now known by tradename TreeNet™
- Breiman introduced RandomForests™ in 2001

- Core idea: multiple trees might be more accurate than single trees
 - By averaging insights of different trees can gain in performance
- Main challenge: how to get multiple trees

Simple methods to generate multiple trees

- Select different random samples for each tree
- Trees are likely to be at least a little different from each other
- Averaging predictions can help “stabilize” results
- Trees work like a committee that votes
 - Say we grow 100 trees. Each tree votes YES or NO on prediction “Claim in next year” or “Large claim in next year”
 - Each policy obtains a number of votes. The larger the number of votes the greater our estimate of the probability of a claim

Multi-tree Methods: Simplest Case

- Simplest example:
 - Grow a tree on training data
 - Find a way to grow another different tree (change something in set up)
 - Repeat many times, eg 500 replications
 - Average results or create voting scheme. Relate PD to fraction of trees predicting default for a given case
- Beauty of the method is that every new tree starts with a **complete** set of data.
- Any one tree can run out of data, but when that happens we just start again with a new tree and **all** the data.



Prediction

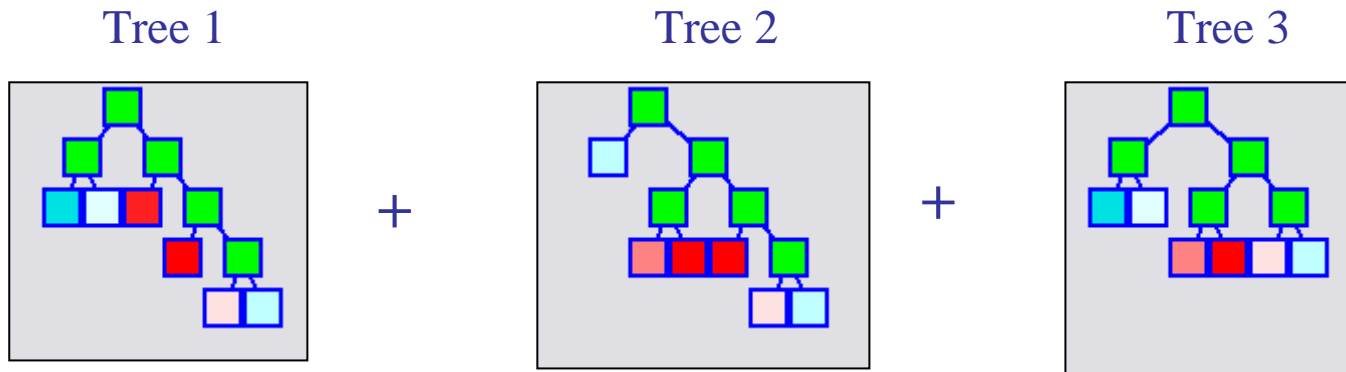
TreeNet™ (aka MART)

- We focus on TreeNet because
 - It is the method used in the real world studies we report here
 - We have found it to be more accurate than the other methods
 - Has placed first in at least two major data mining competitions
 - Building blocks are SMALL trees regardless of training sample size
 - In small samples we have no choice but Treenet *prefers* small trees
 - Very strong resistance to errors in the data including “mislabeled target”
 - Mislabeled target will occur
 - When GOOD/BAD analysis is conducted on relatively new accounts and thus many BADs appear to be GOOD
 - In fraud studies where not all fraud is properly identified. Some fraud confused with legitimate default

TreeNet Process

- Begin with a very small tree as initial model
 - Could be as small as ONE split generating 2 terminal nodes
 - Typical model will have 3-5 splits in a tree, generating 4-6 terminal nodes
 - Output is a probability (eg of default)
- Compute “residuals” for this simple model (prediction error) for every record in data
- Grow second small tree to predict the residual derived from first
- New model is
 - $\text{Tree}_1 + \text{Tree}_2$
- Compute residuals from this new 2-tree model and grow 3rd tree to predict these revised residuals

TreeNet: Trees incrementally revise predicted scores



First tree grown on original target. Intentionally “weak” model

2nd tree grown on residuals from first. Predictions made to improve first tree

3rd tree grown on residuals from model consisting of first two trees

Every tree produces at least one *positive* and at least one *negative* node. Red reflects a relatively large positive and deep blue reflects a relatively negative node. Total “score” is obtained by finding relevant terminal node in every tree in model and summing across all trees

Automated Multiple Tree Generation

- Earliest multi-model methods recommended taking several good candidates and averaging them. Examples considered as few as 3 trees.
- Too difficult to generate multiple models manually
- How do we generate different trees?
 - Bagger: random re-weighting of the data via bootstrap resampling
 - Reweight at random and regrow. Every repetition independent of others
 - RandomForests: Random splits. Tree itself is grown at least partly at random
 - Boosting: weighting based on prior success in correctly classifying a case. High weights on difficult to classify cases
 - Reweighting depends on how successfully a record was previously classified
 - TreeNet: Boosting with refinements. Each tree attempts to correct errors made by predecessors
 - Each tree is linked to predecessors. Like a series expansion where the addition of terms progressively improves the predictions

TreeNet (aka MART)

- We focus on TreeNet because
 - It is the method used in the real world studies we report here
 - We have found it to be more accurate than the other methods
 - Has placed first in at least two major data mining competitions
 - Building blocks are SMALL trees regardless of training sample size
 - In small samples we have no choice but Treenet *prefers* small trees
 - Very strong resistance to errors in the data including “mislabeled target”
 - Mislabeled target will occur
 - When GOOD/BAD analysis is conducted on relatively new accounts and thus many BADs appear to be GOOD
 - In fraud studies where not all fraud is properly identified. Some fraud confused with legitimate default

TreeNet Process

- Begin with a very small tree as initial model
 - Could be as small as ONE split generating 2 terminal nodes
 - Typical model will have 3-5 splits in a tree, generating 4-6 terminal nodes
 - Output is a probability (eg of default)
- Compute “residuals” for this simple model (prediction error) for every record in data
- Grow second small tree to predict the residual derived from first
- New model is
 - $Tree_1 + Tree_2$
- Compute residuals from this new 2-tree model and grow 3rd tree to predict these revised residuals

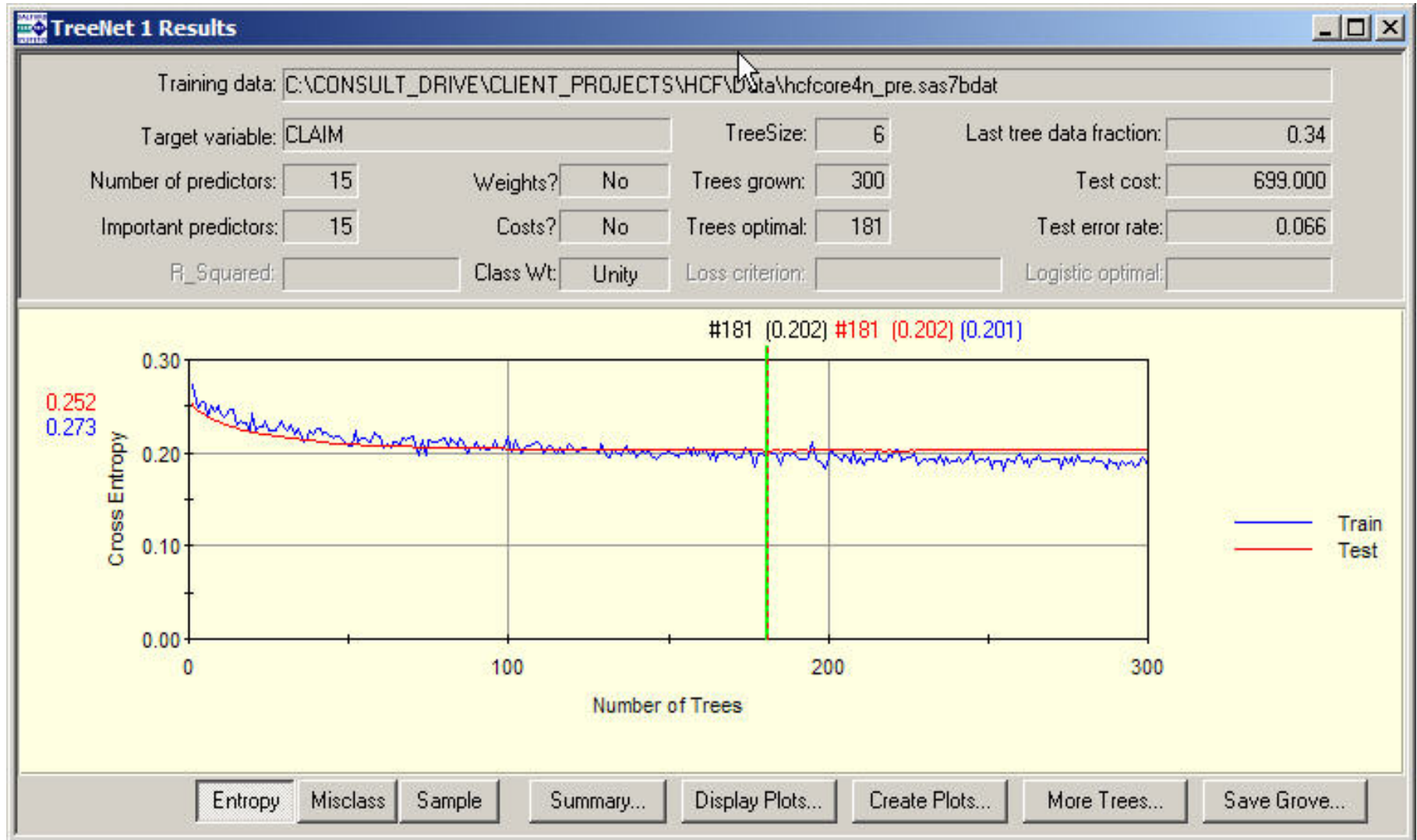
TreeNet methodology: Key points

- Trees are kept small
- Updates are small (downweighted). Like a partial adjustment model. Update factors can be as small as .01, .001, .0001 or even smaller. This means that the model prediction changes by very small amounts in each training cycle
- Use random subsets of the training data in each cycle. Never train on all the training data in any one cycle
- Highly problematic cases are IGNORED. If model prediction starts to diverge substantially from observed data, that data will not be used in further updates
- Cross-validation used for self-test in small data sets
- Model can be tuned to optimize
 - Area under the ROC curve
 - Logistic likelihood (deviance)
 - Classification Accuracy
 - Lift achieved in a specified percentile of the predicted-probability ranked data

Why does TreeNet work?


- Slow learning: the method “peels the onion” extracting very small amounts of information in any one learning cycle
- TreeNet can leverage hints available in the data across a large number of predictors, making use of many of them
- TreeNet self-protects against errors in the dependent variable (vital for fraud studies). If a record is actually a “1” but is misrecorded in the data as a “0” and TreeNet recognizes it as a 1 it will not attempt to get this record correct.
- Can capture substantial nonlinearity and complex interactions

TreeNet Summary Display



TreeNet Model Optimization Criteria: TEST Data

	CXE	ClassError	ROC	Lift
Optimal N Trees:	210	37	83	100
Criterion	0.2038	0.0615	0.8080	4.3120

Actual Class	Total Cases	Percent Correct	0 N=7772	1 N=3577
0 	10,535	71.55	7,538	2,997
1	814	71.25	234	580

TreeNet Balanced

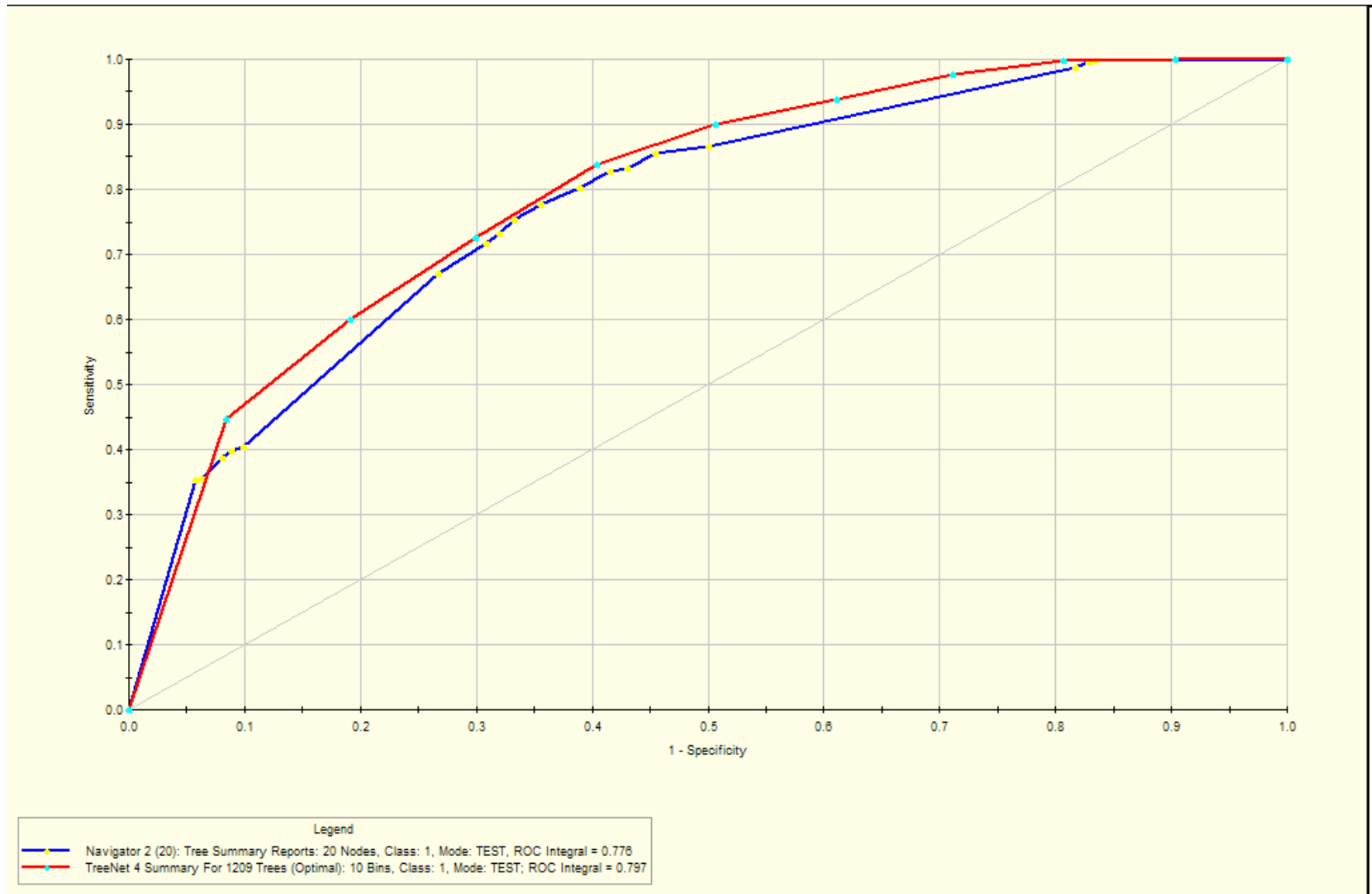
Actual Class	Total Cases	Percent Correct	0 N=6956	1 N=4393
0	10,535	64.47	6,792	3,743
1	814	79.85	164	650

TreeNet Refined

Actual Class	Total Cases	Percent Correct	0 N=6122	1 N=5227
0	10,535	56.84	5,988	4,547
1	814	83.54	134	680

CART

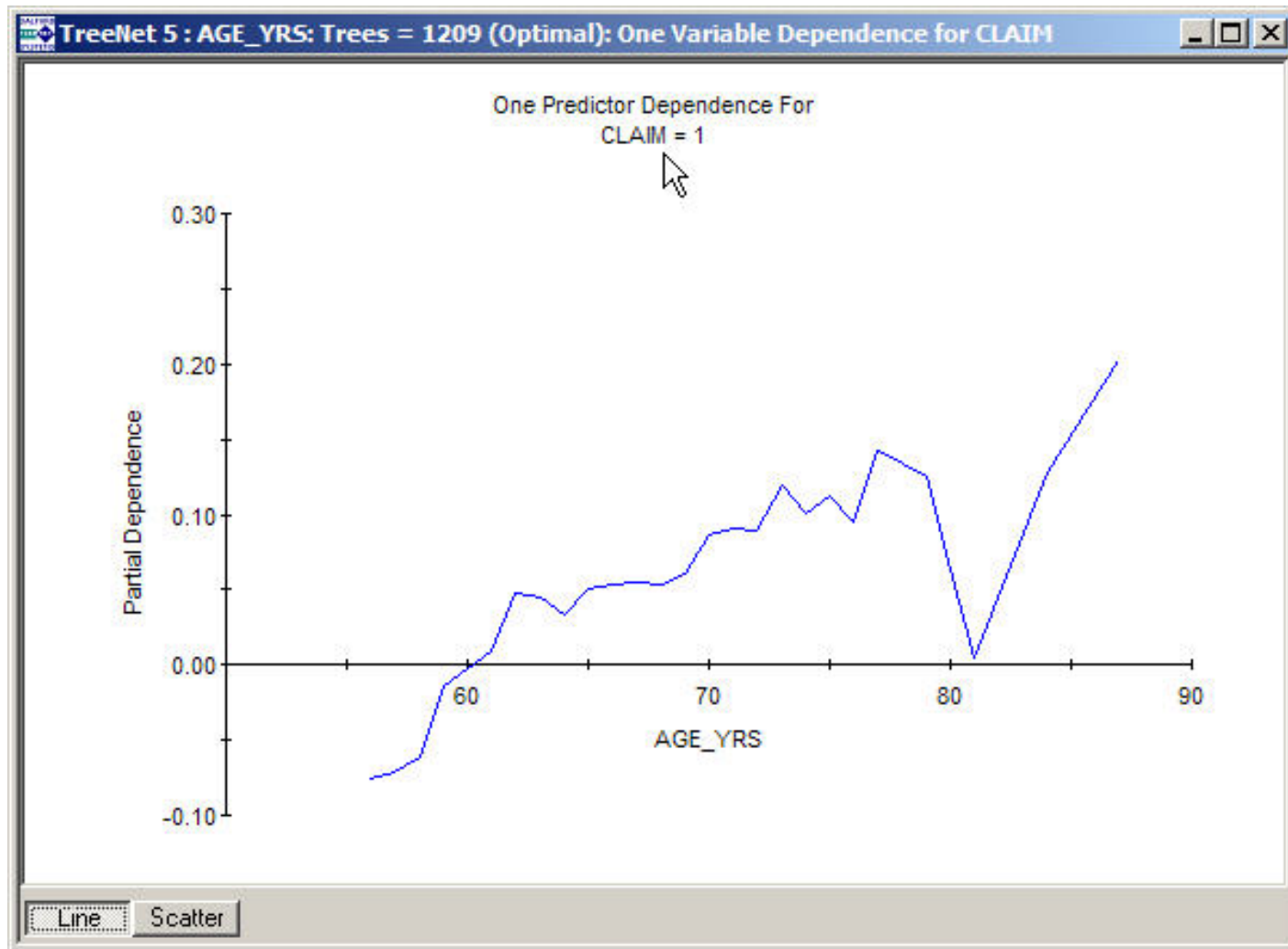
TreeNet vs Unconstrained CART: Test Data



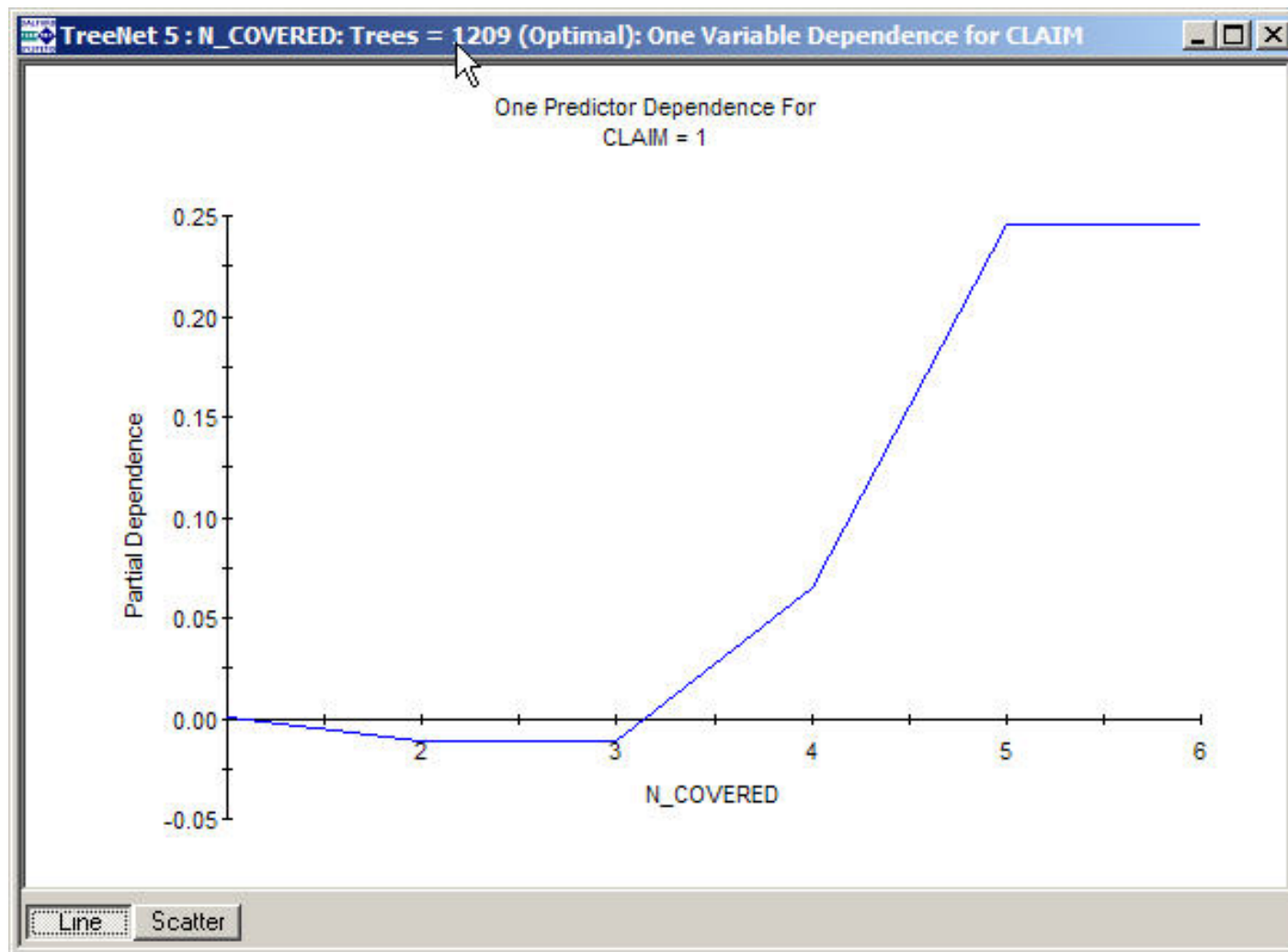
TreeNet Graphs: Dependency Plots

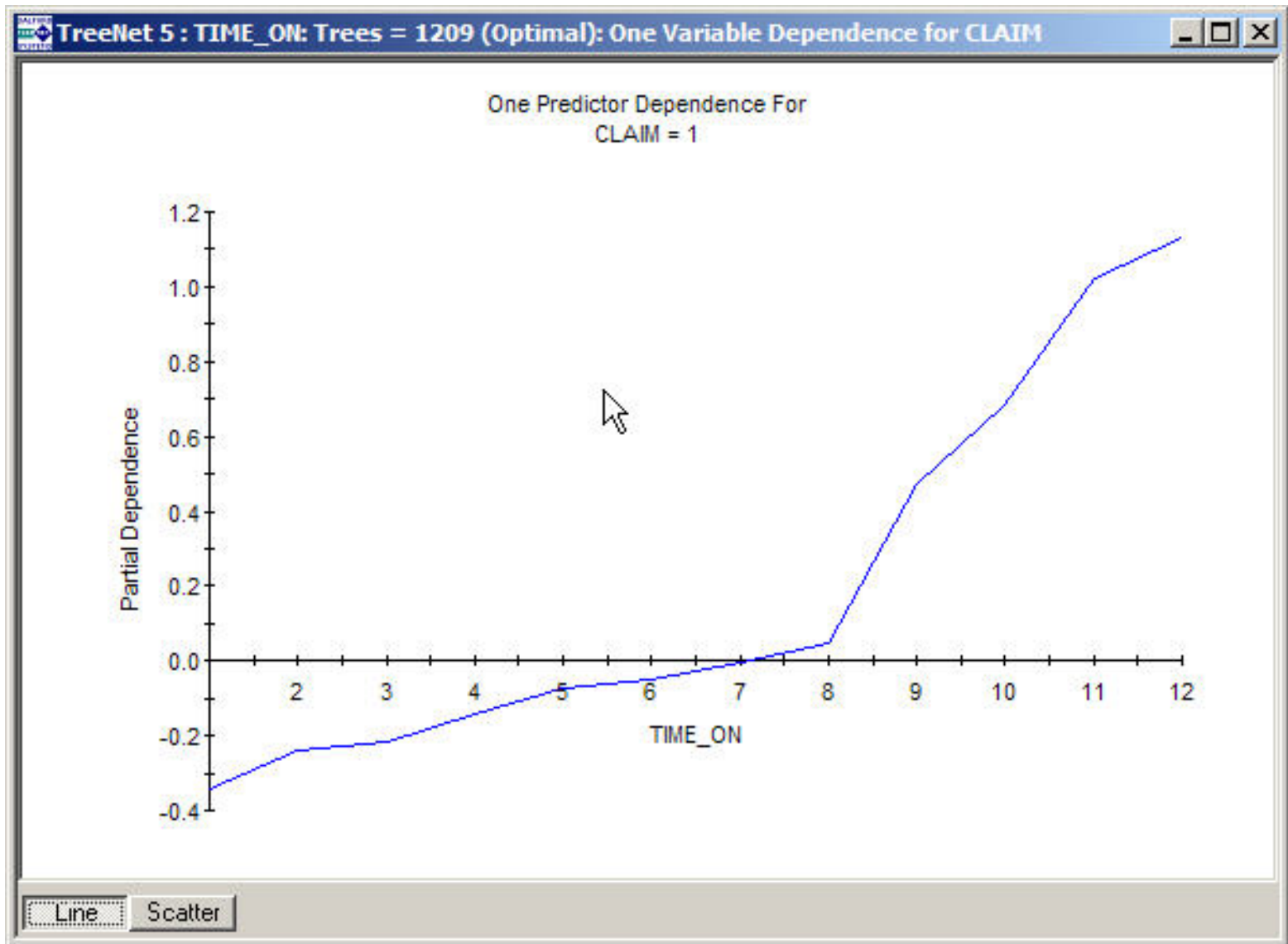
- For every predictor TreeNet produces a “Dependency Plot” which exhibits how varying predictor values influence the outcome (Prob of claim)
- To explore interactions TN produces 3D graphs displaying the response surface above any pair of predictors
- Slices of the 3D graph allow quick detection of important interactions. If the graph remain roughly unchanged in each panel there is no interaction.

TreeNet Dependency Plots

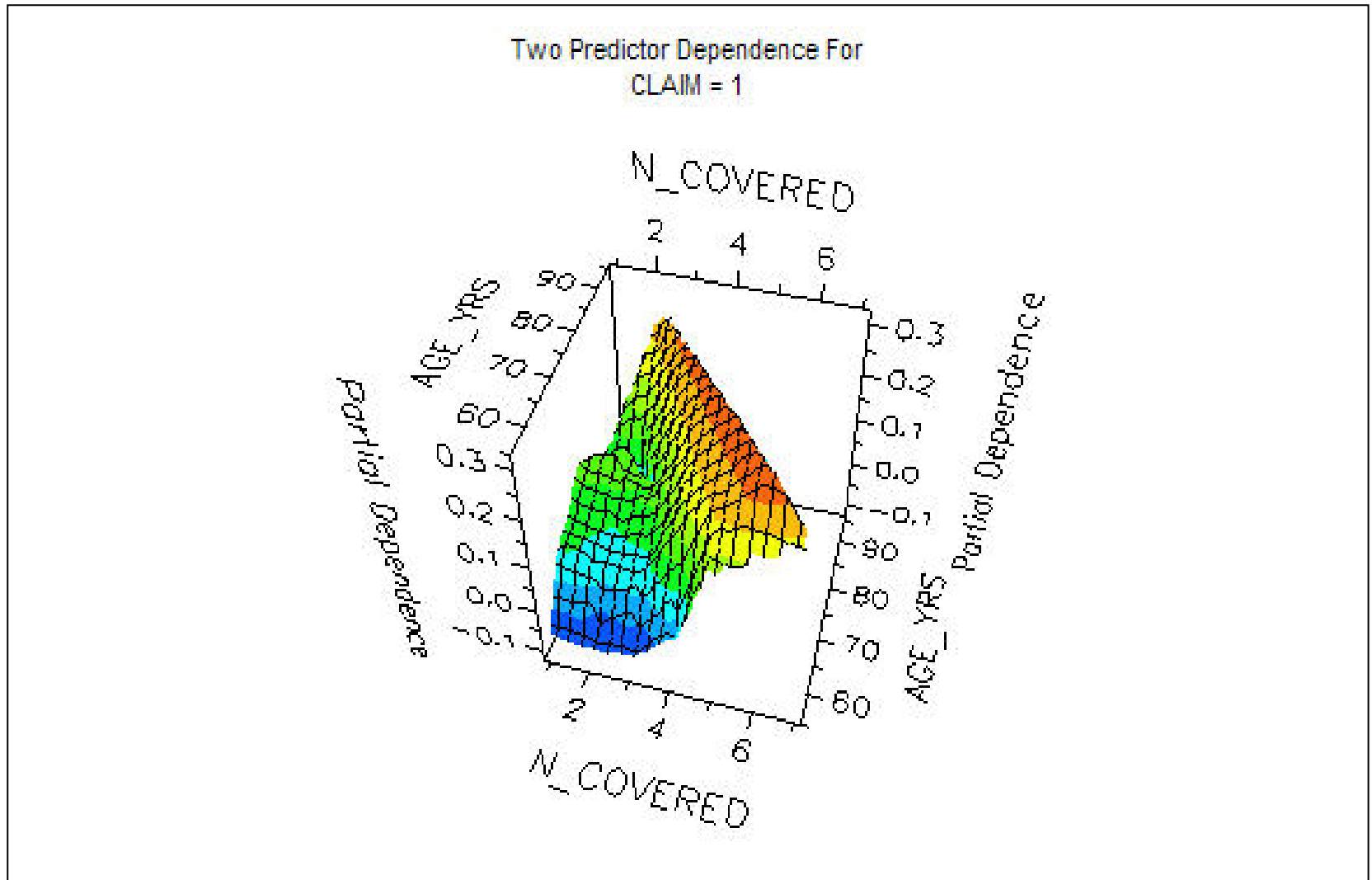


TreeNet Dependency plots

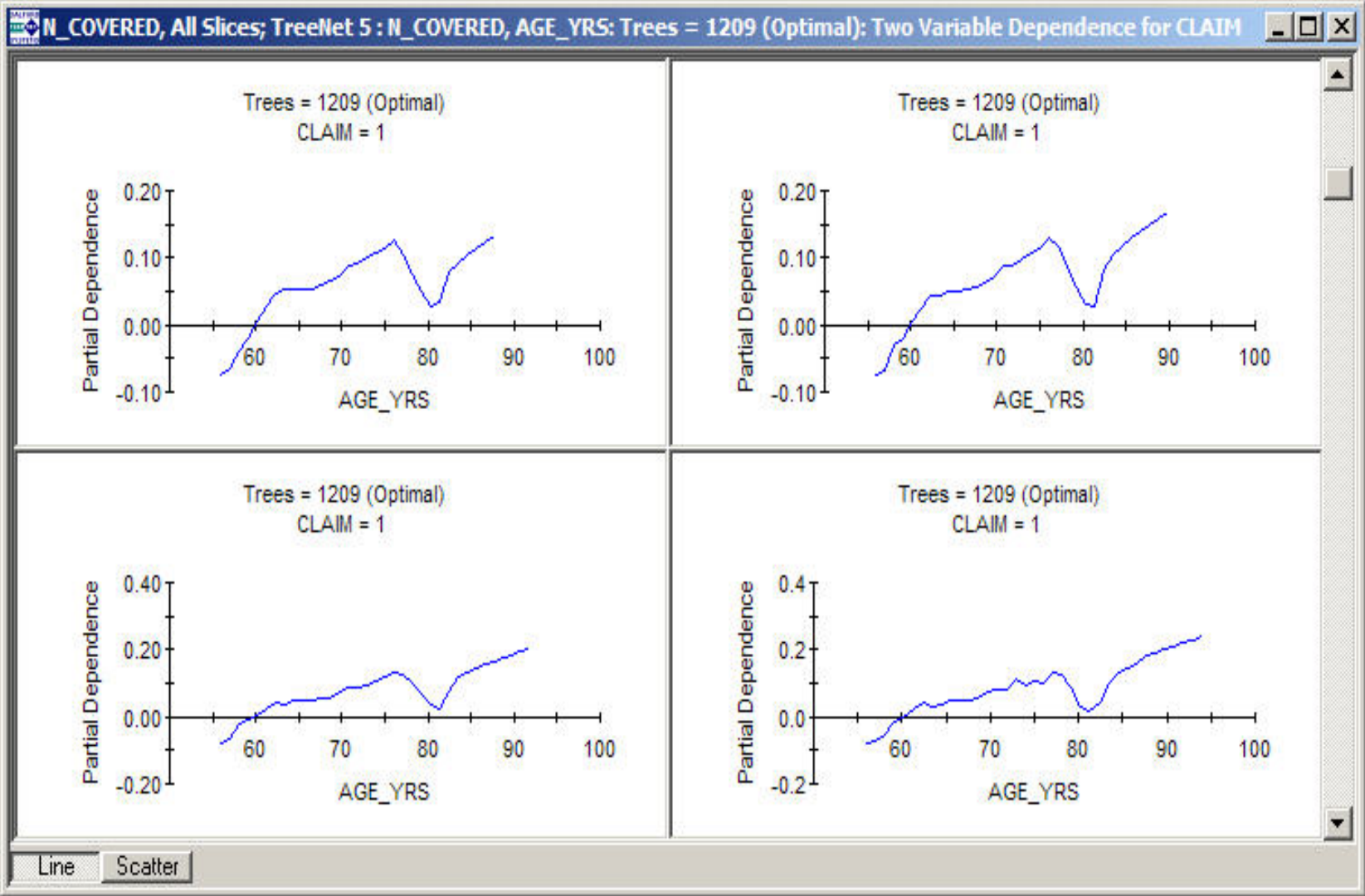




3D graph: AGE_YRS and N_COVERED joint effect on Prob Claim



Examining Interaction Via Slices



Summary

- CART a powerful tool for exploring data, working with imperfect data, assisting in the development of GLMs via interaction discovery and hybrid models
- New developments in CART assist even further in model assessment and model control
- Multi-tree methods offer a next-generation of tools with new graphical outputs

Who is using these methods in actuarial studies?

- See papers on our conference website

<http://www.salforddatamining.com>

- Previous conferences list actuarial track papers available on request
- Consultants involved in actuarial data mining using the tools discussed here include:
 - PriceWaterHouseCoopers
 - EMB
 - Towers Perrin
 - Others, including smaller consultants are investing rapidly in the area

References

- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123-140.
- Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In L. Saitta, ed., *Machine Learning: Proceedings of the Thirteenth National Conference*, Morgan Kaufmann, pp. 148-156.
- Friedman, J.H. (1999). Stochastic gradient boosting. Stanford: Statistics Department, Stanford University.
- Friedman, J.H. (1999). Greedy function approximation: a gradient boosting machine. Stanford: Statistics Department, Stanford University.
- Hastie, T., Tibshirani, R., and Friedman, J.H (2001). *The Elements of Statistical Learning*. Springer-Verlag.