Kass Adjustments in Decision Trees on Binary/Interval Target Variable

Manoj Kumar Immadi

Oklahoma State University, Stillwater, OK

ABSTRACT

Kass adjustment maximizes the independence between the two branches after the split. But how will these adjustments work on interval and binary target variable is a big question? This paper describes split search algorithm in decision trees for selecting useful inputs, followed by comparing the decision tree results of binary target variable and an interval target variable and also provides insights to Kass adjustments/Bonferroni correction in decision trees.

INTRODUCTION

Decision trees are one of the most important classification algorithms in the present day data mining and machine learning techniques. Decision trees are simple rules which are easy to explain and interpret hence decision trees are one of the most important predictive modeling algorithm.

Before attempting to understand the Kass adjustment or Bonferroni correction in decision trees it is advisable to understand how decision trees works. Decision tree is one of the primary tools used for predictive modeling methods among regression and neural. When predicting new cases decision tree scores them using simple prediction rules. Decision tree uses Split search algorithm to select the useful inputs. Split search algorithm is explained in detail in later in this paper. Soon after selection of inputs decision tree builds the model on the selected variables and prunes the tree to optimize the complexity.

A simple decision tree can be seen like this:

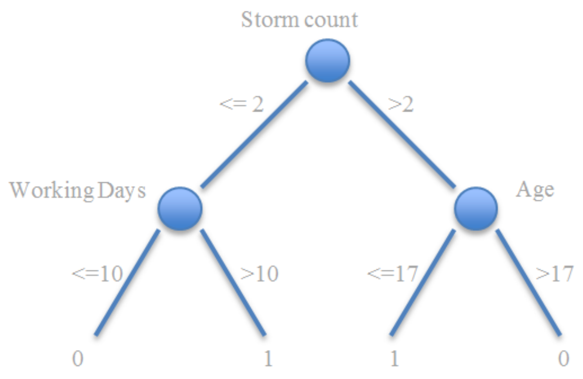


Figure 1. Example for Decision Tree

The above decision tree can be interpreted by means of very simple rules. Target 0 indicates no admit and 1 indicates admit in the sample dataset. Then storm count, age and working days are the input variables to predict the binary admit target variable.

If storm count is greater than 2 and age less than or equal to 17 indicates admits, also storm count less than or equal to 2 and working days greater than 10 also indicates admits. By looking at the decision tree and English rules, we can interpret the decision tree very easily.

KASS ADJUSTMENTS

Before getting into the details it is better to understand how split search algorithm works in Decision trees. To select useful inputs, decision tree use split - search algorithm. Split search starts by selecting an input for creating rule in the training data. If the selected input is interval target variable each and every value serve as a potential split point in the data. If the input is categorical variable average value of the target is taken for each level. Averages will be used as potential split points for the decision tree.

Soon after the selecting the split point two groups of data will be generated. All the records with less than the value of the split point is on left side and the records with greater than the value of the split point are on the right hand side. These two groups forms 2x2 contingency table assuming that the target is a binary variable. Pearson chi-squared statistic is used to quantify the independence. Large value of chi squared statistic indicates that proportion of zeros and ones in two branches is different. Large difference indicates it is a good split. Since the p values are very close to 0 log worth will be calculated and used as the statistic.

$$logworth = -\log(chi - squared\ p - value)$$

Logworth must exceed threshold value in order to a split occurrence. By default the threshold value is 0.7 in enterprise miner.

The algorithm is much more complex than explained above. There are several other factors makes algorithm more complicated includes:

1. Tree settings, such as minimum number of observations for a split and minimum number of observations in leaf and so on.
2. If an input variable is an interval variable number of possible split point might increase since the number increases, the likelihood of obtaining significant value also increases. In this way input with multitude of unique input values has a chance of having large logworth.

Kass adjustments/Bonferroni corrections are based on Boole's inequality, also known as union bound. Boole's inequality says that in any countable set of events. Probability that at least one of those events happens is no greater than the sum of the probabilities of individual events.

For *n* countable set of events $x_1, x_2, x_3 ... x_n$; we have:

$$P(U_n x_n) \leq \sum_n P(x_n)$$

Bonferroni correction is used to counteract the problem of multiple comparisons which is based on Boole's inequality explained above. Statisticians solved the problem of combining the results from multiple statistical tests with the aid of Bonferroni correction.

Since each split point corresponds to a statistical test, Bonferroni corrections are automatically applied to the logworth calculations. These corrections are called Kass adjustments which penalize the inputs with many split points by reducing the logworth. It reduces the logworth to an equal amount of log of the number of distinct input values. These adjustments will allow us for fair comparison of variables with high multitude and low multitude of unique values of input variables. We have special ways handling the data with missing values. Two sets of Kass adjusted logworth values will be calculated. These details are out of scope pertaining to this paper.

Soon after the first split, significance of the secondary and the following splits depends on the significance of the earlier splits. The split search algorithm again goes through a multiple comparison problem. To compensate this algorithm increases the threshold by an amount related to the number of splits. The problem is how these adjustments works on binary and interval target variable.

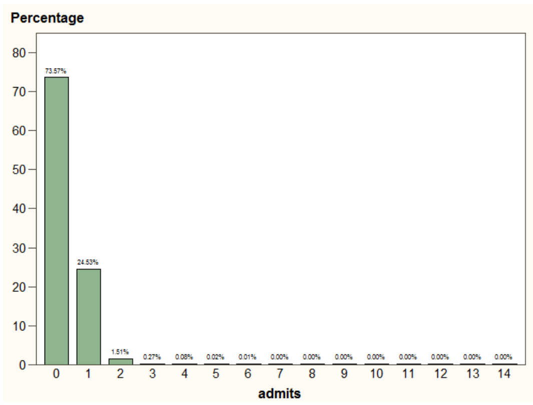
BACKGROUND ON THE DATA

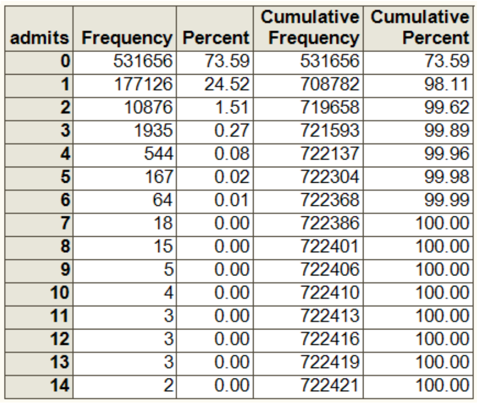
The dataset is related to weather and the health care usage facilities. The dataset is created from 5 different dataset the final dataset contains the information about weather, admits, storm and population in a given particular area code and for each age group. The target variable Admits contains the value of number of admits in a given particular area and in a particular age group and for a particular DRG (type of disease). This paper discusses about the Kass adjustments hence it will not go through each and every detail of data preparation. Few of the important things are mentioned and relevant diagrams are shown below.

Admits is an interval target variable, from Admits a new binary target variable *isAdmit* is created which serves the purpose of comparing the results of interval and binary target variable. Many other new variables are created from the existing variables explaining them is out of context from this paper perspective.

AdmitsProp a new interval target variable created from admits and population is created. The new target variable is created based on the following observations. By comparing the normalized distribution charts of Admits and AdmitsProp the new target variable was used for comparison purposes.

In the Admit target variable 73.59% of observations does not have any admits in the dataset. 24.52% observations have the number of admits as one and the rest 1.89 % observations contains admits ranging from 2 to 14. The distribution of admits is highly right skewed, high number of admits are very rare and mostly occur in densely populated areas. In the Figure 2 we can have look at the number of admits and number of observations pertaining to Admit.





admits	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	531656	73.59	531656	73.59
1	177126	24.52	708782	98.11
2	10876	1.51	719658	99.62
3	1935	0.27	721593	99.89
4	544	0.08	722137	99.96
5	167	0.02	722304	99.98
6	64	0.01	722368	99.99
7	18	0.00	722386	100.00
8	15	0.00	722401	100.00
9	5	0.00	722406	100.00
10	4	0.00	722410	100.00
11	3	0.00	722413	100.00
12	3	0.00	722416	100.00
13	3	0.00	722419	100.00
14	2	0.00	722421	100.00

Figure 2. Admit (target variable) frequency chart

Data is normalized by calculating new target variable AdmitsProp. The new variable is the proportion of admits to the population per 10000 (ten thousand). Even though the distribution of admits proportion is also highly skewed with 99.49% of admits proportion concentrated to 0-60 with few exceptional cases like 2500, 5000, 10000 etc. Reason behind this is high number of admits in low populated areas.

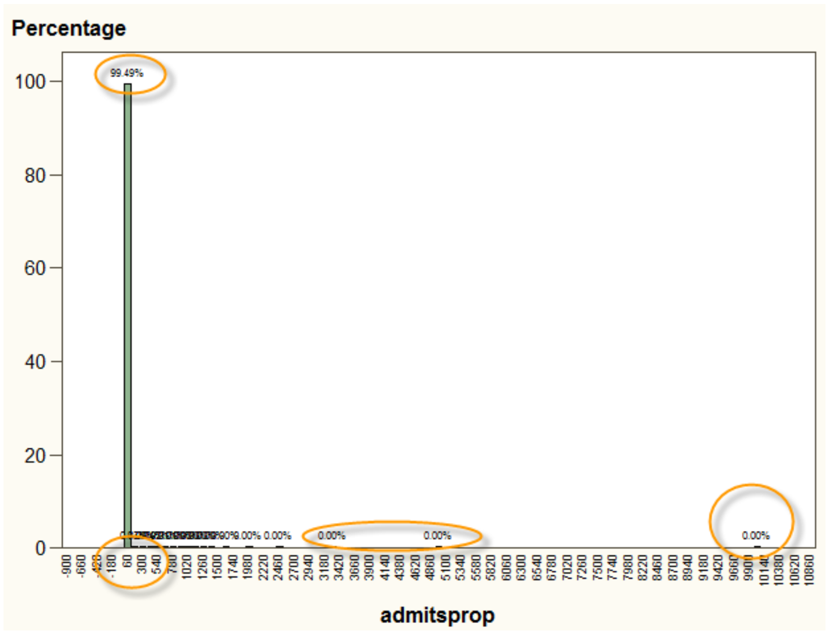


Figure 3. AdmitsProp variable distribution

For better results and for better comparison purposes the outliers were removed before proceeding further. The new interval target variable AdmitsProp contains the values in the range of zero to two hundred only.

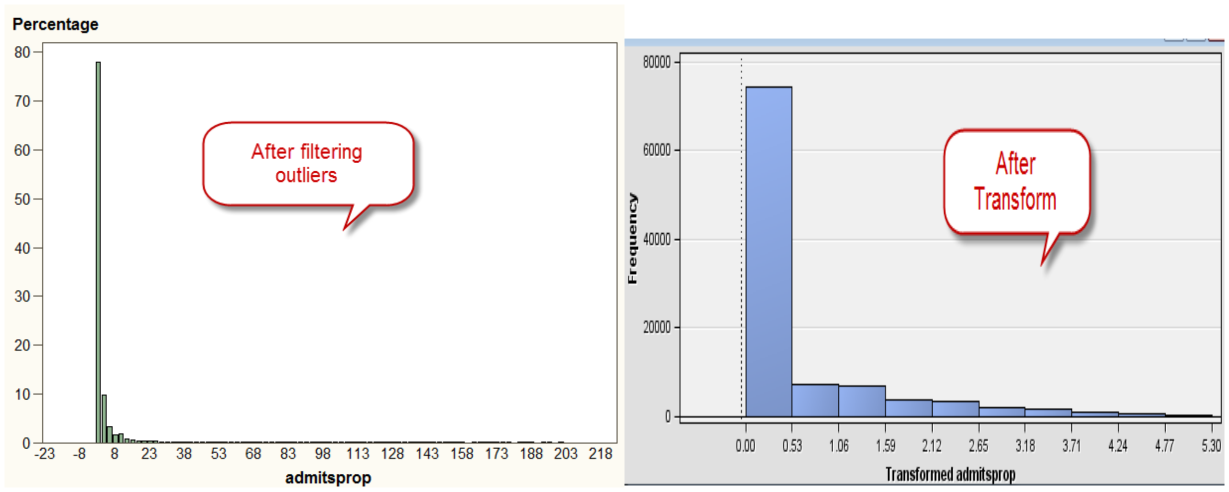
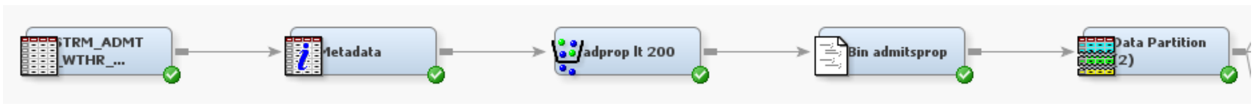


Figure 4. admitsprop after distribution after filtering and transforming

After the preparation of final dataset our first task in this dataset is to look for any inconsistencies, errors or extreme values in the data. Frequency distribution, descriptive statistics, cross tab and multiplot nodes were used in order to analyze and various techniques used for fixing the issues. Few of the important things are noted down below:

- Missing values in storm are replaced with ‘0’ indicates ‘no storms’
- Missing values in weather are replaced by the average value of temperatures in that particular area code and in that particular month.
- Minimum incubation is deleted since every value is 1.
- We can see high number of admits in months 2,3,11 and 12 and we can observe less admits in months 6 and 7. Admits has shown seasonality in the sample data.
- We have less percentage of missing values in sn0-sn5; ah0-ah5; al0-al4; atd0-atd4; dl0-dl4; htd0-htd4; mh0-mh4; ml0-ml4; prcp0-prcp4 and are replaced by the mean.
- AdmitsProp is having only 0.224% of values which are greater than 200, thus decreased skewness in the target variable by eliminating the outliers.
- Different transformations were applied to different variables in order to improve the model performance.

The following is the screen shot of the data preparation:



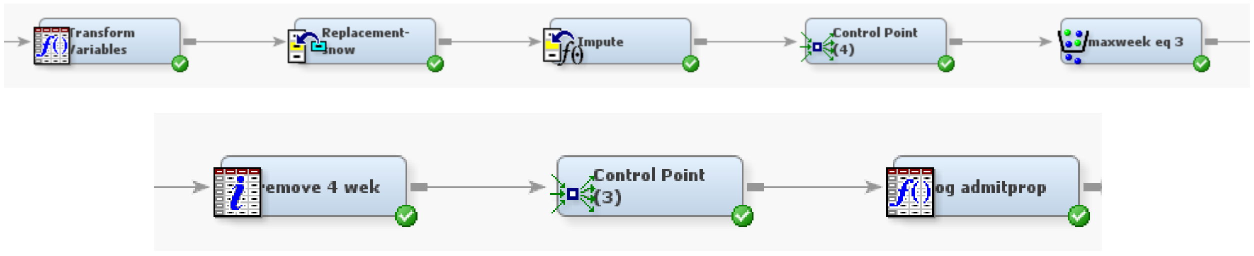


Figure 5. Data Preparation

Data understanding is done by running the following nodes in the enterprise miner:

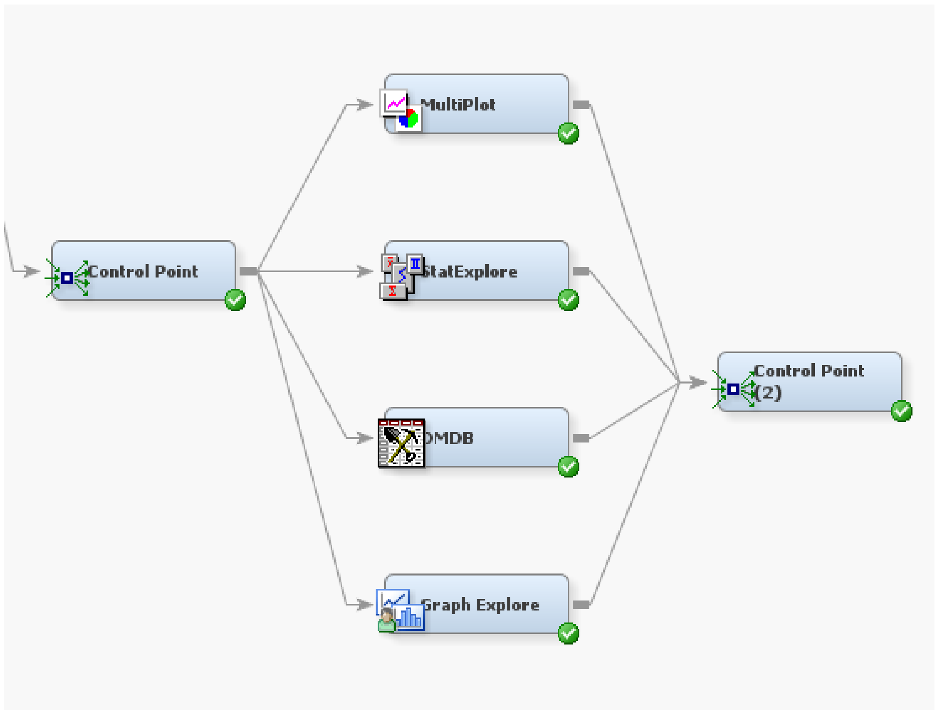


Figure 6. Data Understanding

METHODOLOGY

The comparison Kass adjustments/Bonferroni correction for binary/interval target variable is done based on the following methodology. Decision tree in enterprise comes with various options. Apart from changing the splits number of branches we have various options. For categorical input SAS offers three different spilt worth criteria, they are ProbChiSq, Gini and Entropy it is known that they give similar results if the number of levels in each categorical input is similar. We also have option to turn on/off the Bonferroni Adjustment. By using this option we can compare how these adjustments work on binary/interval target variable.

My goal is to compare the results by turning on and off the Bonferroni Adjustment for binary and interval target variable, thus comparing the results.

The first task is to compare the results of binary target variable with and without Bonferroni Adjustment. The first three nodes Figure 7 are Entropy, Gini and ProbChiSq models with Bonferroni Adjustment and the other three nodes are without Bonferroni Adjustment.

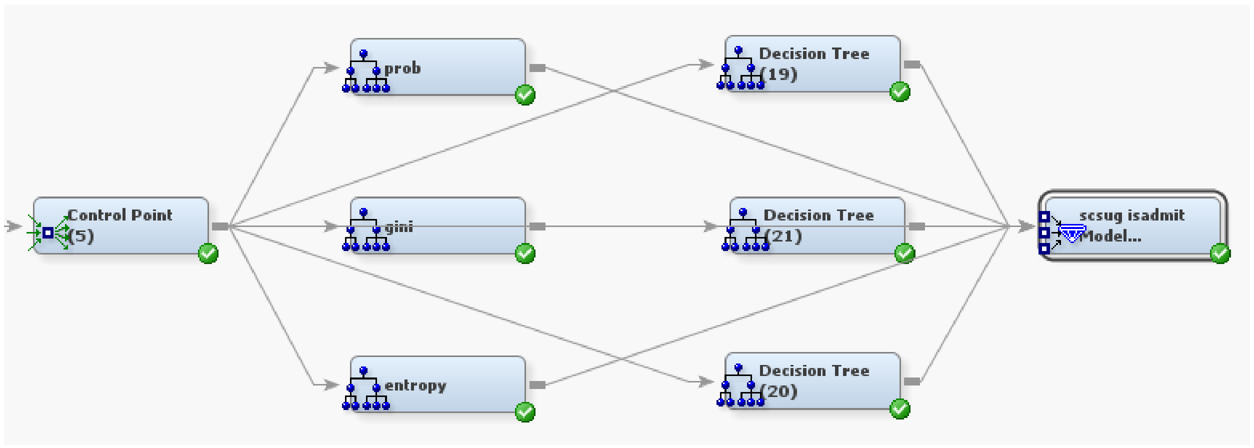
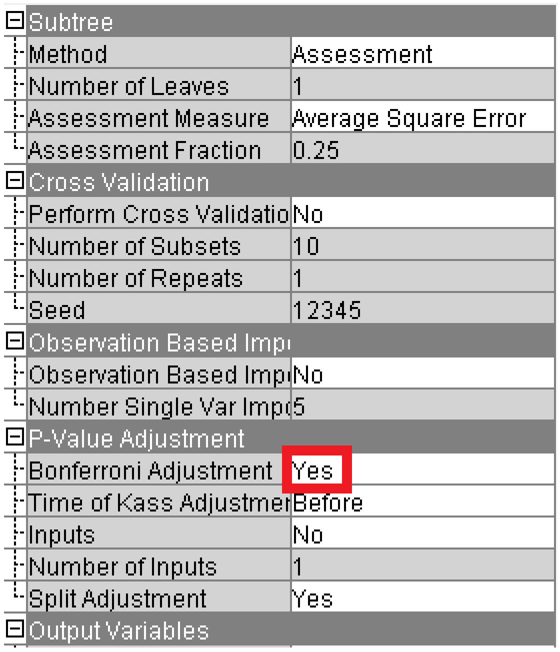


Figure 7. Binary target with/without Bonferroni Adjustment

The ProbChiSq with Bonferroni Adjustment is selected as the best model from the model comparison node. The properties of the selected model are:



Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25
Cross Validation	
Perform Cross Validatio	No
Number of Subsets	10
Number of Repeats	1
Seed	12345
Observation Based Impo	
Observation Based Impo	No
Number Single Var Impo	5
P-Value Adjustment	
Bonferroni Adjustment	Yes
Time of Kass Adjustmer	Before
Inputs	No
Number of Inputs	1
Split Adjustment	Yes
Output Variables	

Figure 8. Properties of ProbChiSq model with Bonferroni Adjustment

Bonferroni Adjustment in the properties panel is changed to no for the other three decision trees.

The next task is to compare the results interval target variable with and without Bonferroni Adjustment. The first three nodes in Figure 9 are Entropy, Gini and ProbChiSq models with Bonferroni Adjustment and the other three nodes are without Bonferroni Adjustment.

Please refer results section for the comparison how Bonferroni Adjustment/Kass adjustments improved the results of binary target variable and interval target variable.

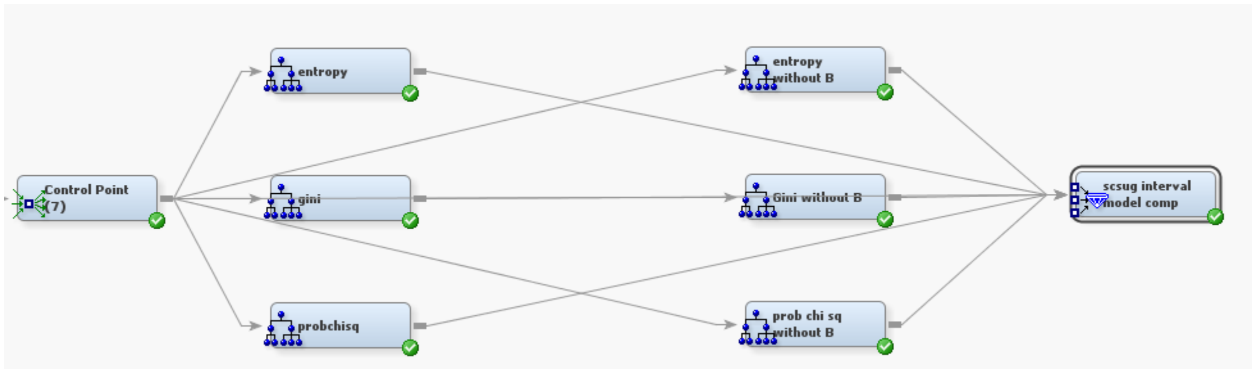
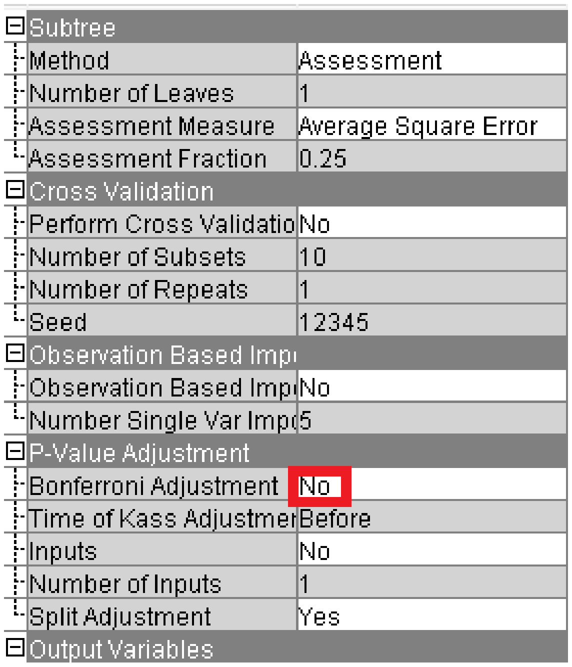


Figure 9. Interval target with/without Bonferroni Adjustment

Properties of ProbChiSq model without Bonferroni Adjustment are shown below:



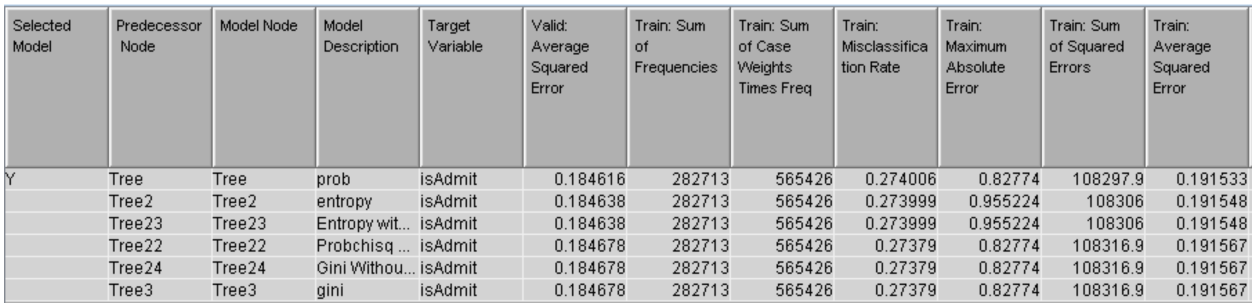
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25
Cross Validation	
Perform Cross Validatio	No
Number of Subsets	10
Number of Repeats	1
Seed	12345
Observation Based Imp	
Observation Based Imp	No
Number Single Var Imp	5
P-Value Adjustment	
Bonferroni Adjustment	No
Time of Kass Adjustme	Before
Inputs	No
Number of Inputs	1
Split Adjustment	Yes
Output Variables	

Figure 10. ProbChiSq properties without Bonferroni Adjustment

RESULTS

For simplification purposes this paper compares the results selected model with/without Bonferroni Adjustment for each binary and interval target variable. Kass adjustments improved the results in all other models too. The models built for comparison purposes are very simple with properties, maximum branch of 2 and maximum depth of 6.

Results of binary target variable are shown below:

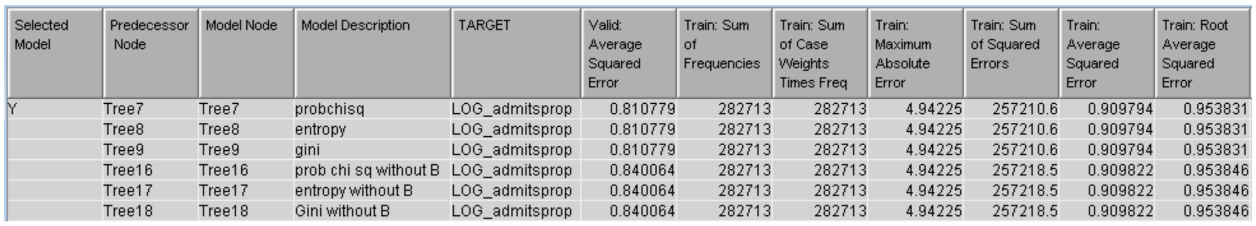


Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Valid: Average Squared Error	Train: Sum of Frequencies	Train: Sum of Case Weights Times Freq	Train: Misclassification Rate	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error
Y	Tree	Tree	prob	isAdmit	0.184616	282713	565426	0.274006	0.82774	108297.9	0.191533
	Tree2	Tree2	entropy	isAdmit	0.184638	282713	565426	0.273999	0.955224	108306	0.191548
	Tree23	Tree23	Entropy wit...	isAdmit	0.184638	282713	565426	0.273999	0.955224	108306	0.191548
	Tree22	Tree22	Probchisq ...	isAdmit	0.184678	282713	565426	0.27379	0.82774	108316.9	0.191567
	Tree24	Tree24	Gini Withou...	isAdmit	0.184678	282713	565426	0.27379	0.82774	108316.9	0.191567
	Tree3	Tree3	gini	isAdmit	0.184678	282713	565426	0.27379	0.82774	108316.9	0.191567

Figure 11. Results of binary target variable

Valid: Average Squared Error is the selection criteria for selecting the model, model comparison selected ProbChiSq with Bonferroni Adjustment as the best model. Valid average squared error of the selected model is 0.184616, but when you look at the train: average squared error used for the model building the average squared error of the selected model with and without Bonferroni are 0.191533 and 0.191567 respectively. Train average squared error has shown significant improvement by including Bonferroni Adjustment. Train average squared error improved 0.000037.

Results of interval target variable are shown below:



Selected Model	Predecessor Node	Model Node	Model Description	TARGET	Valid: Average Squared Error	Train: Sum of Frequencies	Train: Sum of Case Weights Times Freq	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error	Train: Root Average Squared Error
Y	Tree7	Tree7	probchisq	LOG_admitsprop	0.810779	282713	282713	4.94225	257210.6	0.909794	0.953831
	Tree8	Tree8	entropy	LOG_admitsprop	0.810779	282713	282713	4.94225	257210.6	0.909794	0.953831
	Tree9	Tree9	gini	LOG_admitsprop	0.810779	282713	282713	4.94225	257210.6	0.909794	0.953831
	Tree16	Tree16	prob chi sq without B	LOG_admitsprop	0.840064	282713	282713	4.94225	257218.5	0.909822	0.953846
	Tree17	Tree17	entropy without B	LOG_admitsprop	0.840064	282713	282713	4.94225	257218.5	0.909822	0.953846
	Tree18	Tree18	Gini without B	LOG_admitsprop	0.840064	282713	282713	4.94225	257218.5	0.909822	0.953846

Figure 12. Results of interval target variable

Valid: Average Squared Error is the selection criteria for selecting the model, model comparison selected ProbChiSq with Bonferroni Adjustment as the best model. Valid average squared error of the selected model is 0.810779, but when you look at the train: average squared error used for the model building the average squared error of the selected model with and without Bonferroni are 0.909794 and 0.909822 respectively. Train average squared error has shown significant improvement by including Bonferroni Adjsutment. Train average square error is improved 0.000028.

CONCLUSION

Kass adjustments showed some significant improvement in both binary and interval target variable. But it has shown better improvement in binary target variable than the interval target variable.

REFERENCES

[1] “Using Decision Trees to Identify Medicare Part B Providers for Audit”, poster SESUG - 2002, by Noel McKetty, First Coast Service Options, Jacksonville, FL; Donna Mohr, University of North Florida, Jacksonville, FL; “http://analytics.ncsu.edu/sesug/2002/PS08.pdf”

[2] “Decision Trees”, by Andrew W. Moore ProfessorSchool of Computer ScienceCarnegie Mellon University “http://www.autonlab.org/tutorials/dtree18.pdf”

[3] “Decision trees - what are they?” SAS support community, “http://support.sas.com/publishing /pubcat/ chaps/57587.pdf”

[4] “K Nearest neighbors Classifier & Decision Trees”, “http://www.ibms.sinica.edu.tw/~pan/ classification /documents/KNN&DECISION%20TREE.ppt”

[5] “Bonferroni Inequalities”, “http://mathworld.wolfram.com/BonferroniInequalities.html”

[6] “Proof of Booles inequality”, “http://planetmath.org/encyclopedia/ProofOfBooleInequality.html”

ACKNOWLEDGEMENTS

Thanks for IICH and Analytics2011 team to work on this dataset.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Manoj Immadi, Oklahoma State University, Stillwater, OK, Email: manoj.immadi@okstate.edu

Manoj Immadi is a master’s student in Management Information Systems at Oklahoma State University. He has two years of professional experience as programmer analyst. He is SAS Certified Advanced Programmer for SAS® 9 and Certified Predictive Modeler Using SAS® Enterprise Miner 6.1