CS4341 Project 3 - Report

Connect4 Decision Tree By Alex Bennett and Paulo Carvalho

Table of Contents

Feature Description, Design and Preprocessing

The developed software creates 5 categories of features each of which is composed of different types. This includes: hasWin, determines if there is a win on the board; calcRows, counts number of pieces per player per row; calcCols, counts number of pieces per player per column; neighborWeights, determines value of board based on value of pieces as determined by their neighbors; hasConnect, determines if a given player has two or three pieces together. Notice that all statistics below are marked as either VALID or INVALID. This refers only to it passing the preprocessing stage. Some of the heuristics were eliminated later on during kappa analysis. Notice that the heuristics here mentioned, were designed with two perspectives. First, most of the heuristics were designed with no prior knowledge and serve

as benchmarking for the WEKA results. This was deemed important since the team has no prior experience with the software. Second, some of the heuristics use of prior knowledge and attempt to capture notions of the game that would be used by an actual human player.

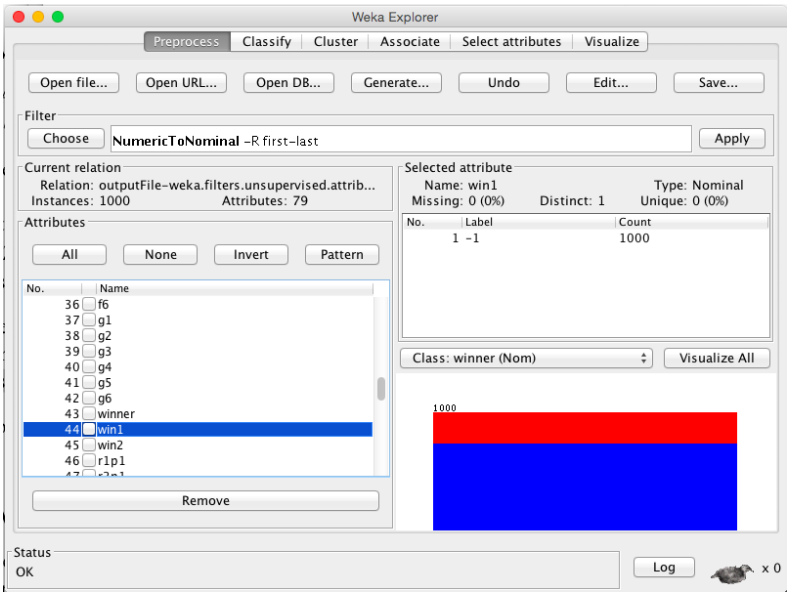
HasWin

Description

One of the most elementary heuristics, it returns the value of the winning player, else it returns -1 if none of the players have won.

Analysis - INVALID

This feature was eliminated by inspection during the preprocessing. This decision was based on the observation that both “win1” and “win2” yield a single (and the same) value given that none of the 1000 boards contained a win. The WEKA screen for this feature split by player 1 won (blue) and player 2 won (red) is shown below.



ConnectN

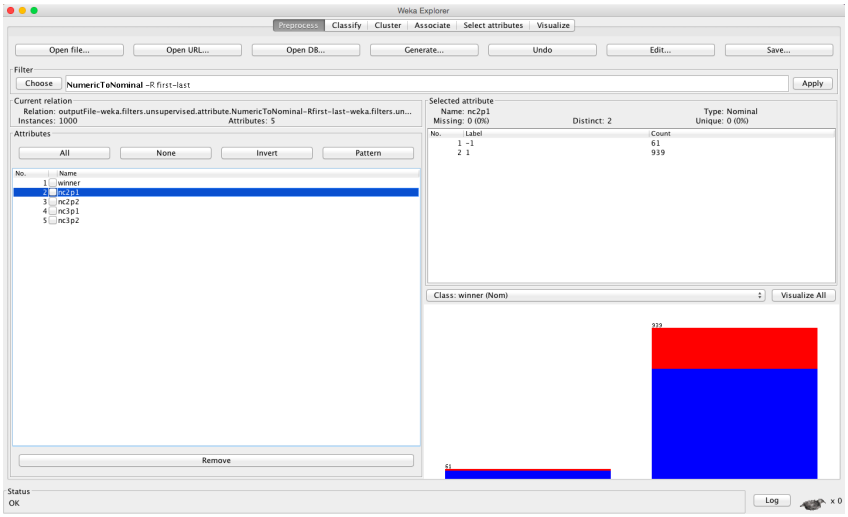
Description

This heuristic takes as input a board and which player is computing the heuristic and returns a value of their player number (1 or 2) if the player has at least one set of N pieces together or -1 if the given player does not have the given number of connects. For determining which heuristics are of most value, an attempt was made at being neutral and diverse. As such the heuristic is computed for 2 and 3 connects for both player 1 and 2.

Analysis - VALID

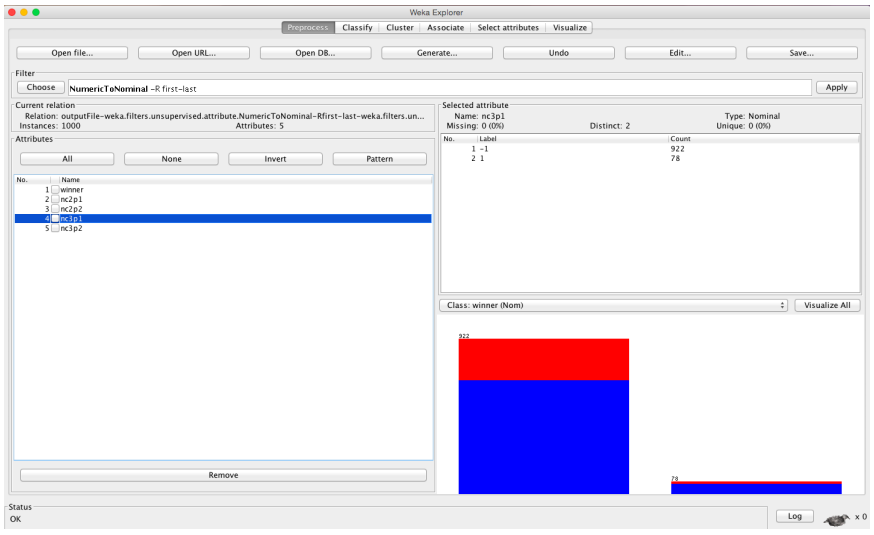
Connect-2

Although having a small alteration on the proportion of wins to losses (WEKA would benefit from showing percent contribution on its graphs… It is sometimes hard to inspect them), the feature does contribute towards a classification as either win or lose.



Connect-3

It was also noticed that having a connect-3 made it more likely that a given player won the game. Although not determined from preprocessing, this heuristic may benefit from either a more discrete approach in which the number of connects are counted or from marking if a player has a connect when the other player does not.



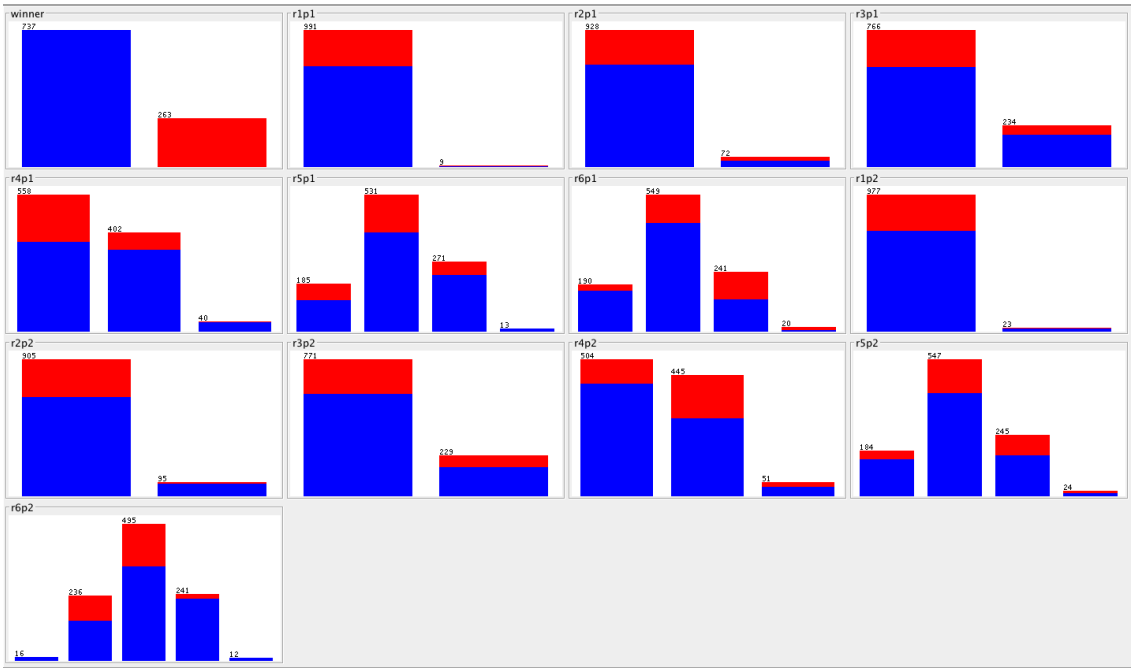
CalcRows

Description

Calculates the number of pieces that each player has on each row. This heuristic results in a total of 12 heuristic values which are appended in the csv file with name “rnpx”.

Analysis - INVALID

As shown in the figure below, no sensible forecast can be drawn by attempting to relate the number of pieces a player has in a row to the results of the game.



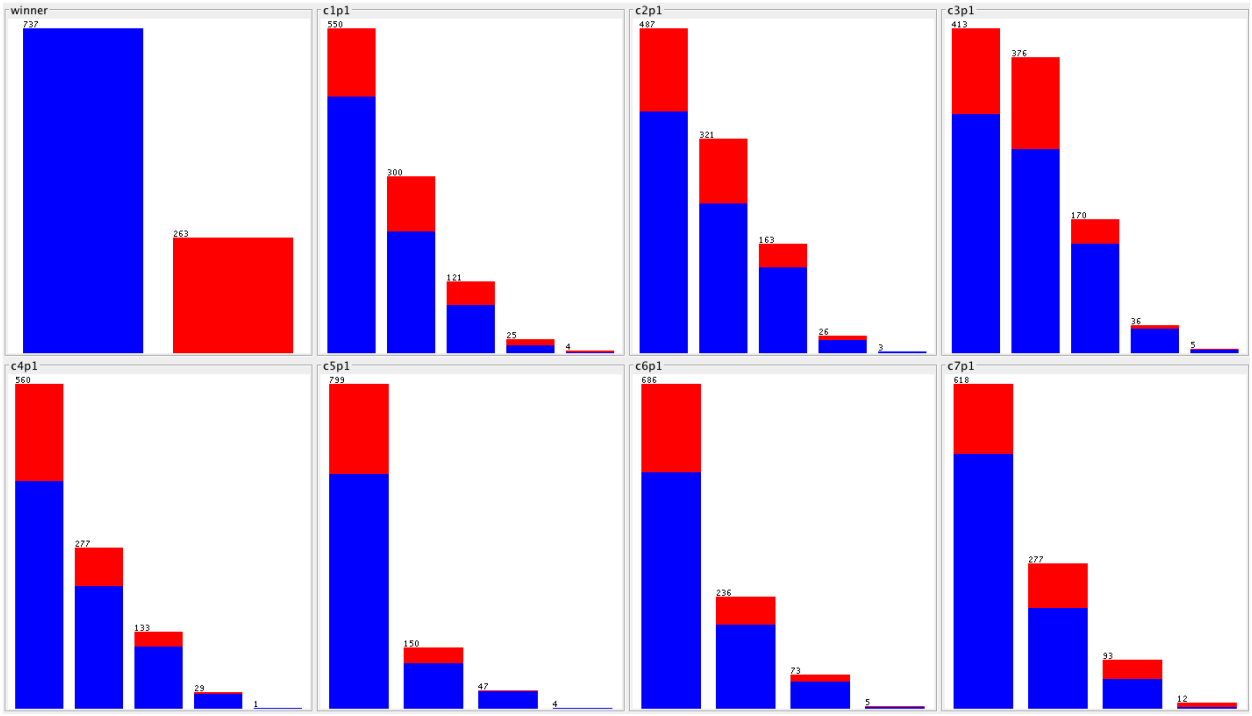
CalcCols

Description

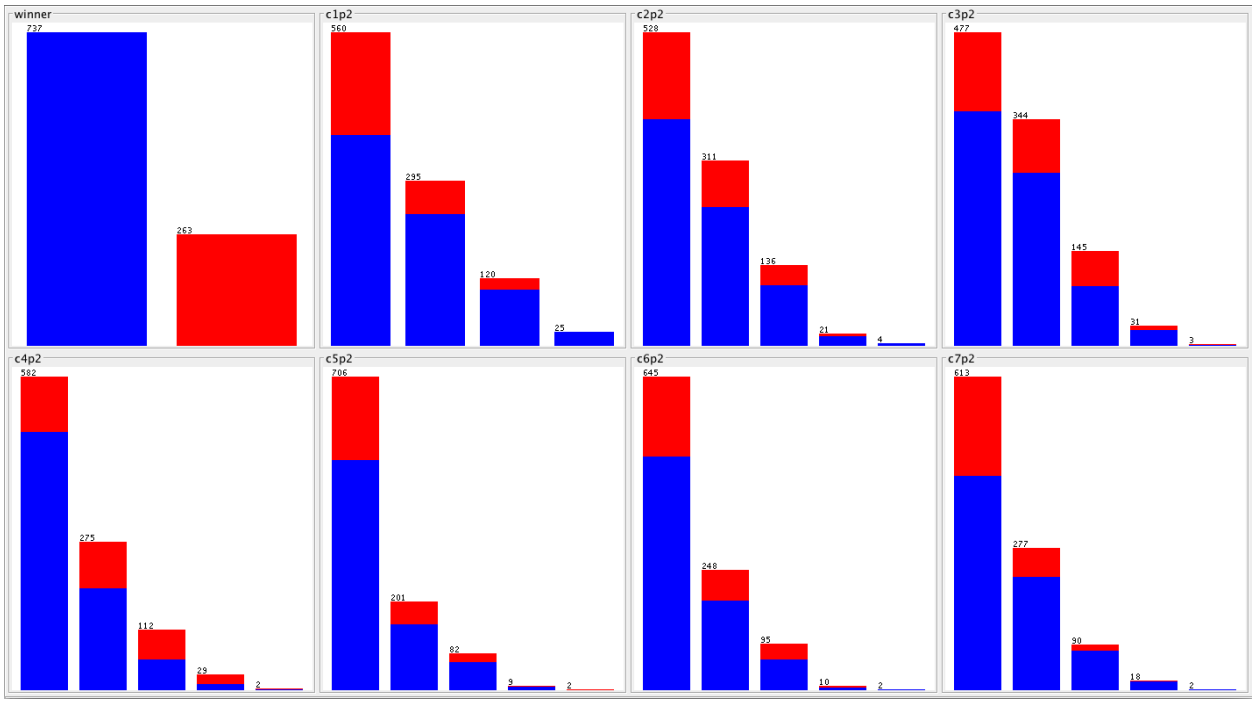
This heuristic returns the number of pieces that each player has on each column of the board. Notice that the team considered narrowing down this heuristic to return a single number based on the proximity of a given player’s pieces to the center column given that prior knowledge dictates that playing near or at the center column yields the best chances of winning. However, in an attempt to remain neutral and to allow the decision tree to show its full potential, it was decided against narrowing the heuristic and instead allowing information on all columns.

Analysis - VALID

The figure below shows the plots for the number of pieces player 1 has on each of the columns colored by final win state. Notice that, as expected, player 1 appears to win more often when he has more pieces in the center columns.



The figure below contains information about player 2 in the same formatting as shown above. A similar conclusion can be drawn, in that the more pieces your enemy has in one of the center column the better is your expected outcome.

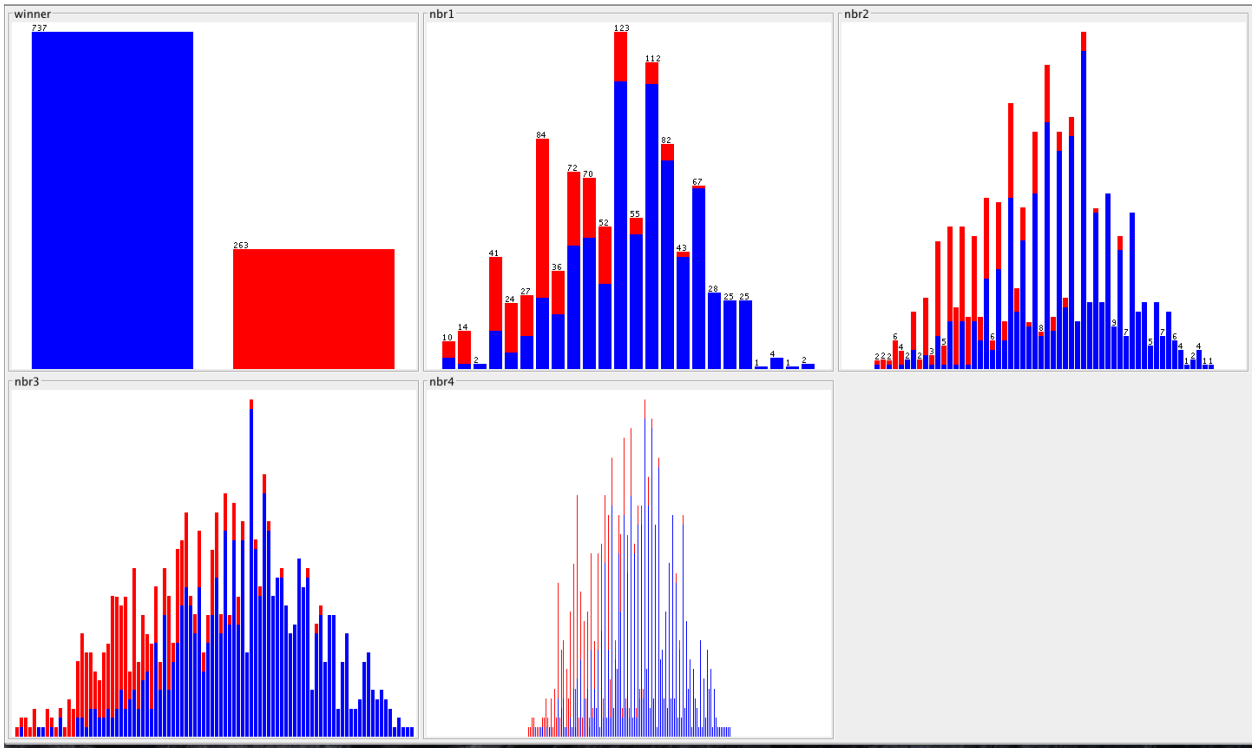


NeighborWeights

This heuristic was developed by the team for project 1 of the class. This heuristic can be calculated for the pieces of either player and takes as argument the radius of calculation. It works by computing for each piece the sum of the values of all its neighbors. Where a neighbor of the same kind adds one to its value whilst an enemy neighbor subtracts one from its value (empty slots have no effect). The final value returned from the heuristic is the sum of the values in all positions of the board. For a more complete description see the report for project 1. In developing this heuristic the team attempted to summarize in a single number the overall maneuverability of the player. If the board scores high it is likely that the player has several unblocked pieces that are near each other and is (hypothesis) therefore more likely to score a win.

Analysis - VALID

As expected, the preprocessing clearly reveals that this heuristic is valid for determining which player should be victorious. As can be noticed from the graphs below where blue corresponds to player 1 winning and red corresponds to player 1 defeat, A high value of the heuristic (as graph tends to right) correlates strongly with a victory of player 1.



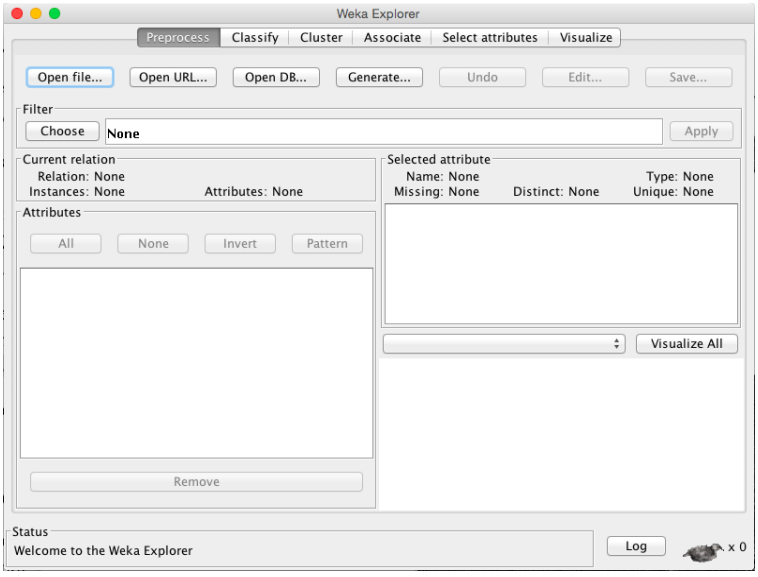
WEKA Procedure

File Importing and Workspace Setup

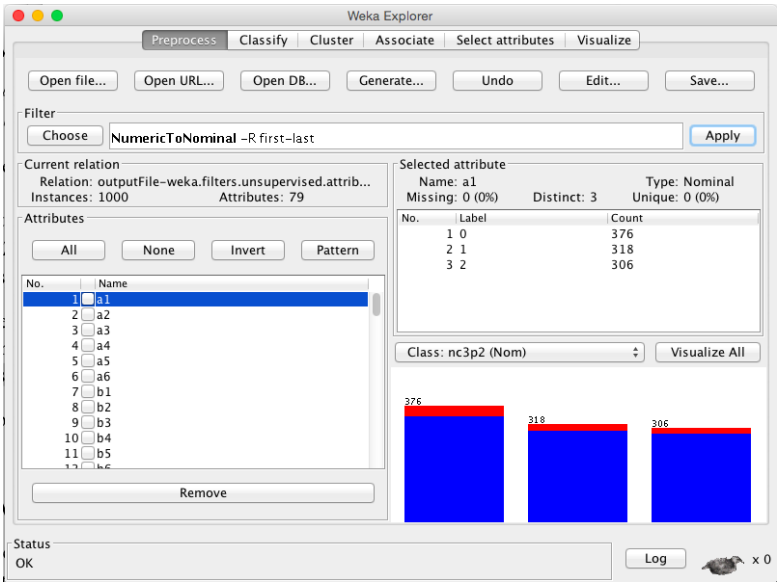
1. A version of WEKA compatible with the team's operating system was installed.

2. Upon opening WEKA a window with four options was displayed and “Explorer” was selected.

3. The option for “open file…” was then selected on the new menu (as shown below) leading to a dialog box for selecting the file containing the data set with the appended statistics.



4. On the same menu the filter was chosen by selecting “choose” and navigating to weka→ filters→ unsupervised→ attribute→ NumericToNominal. The final window with the file imported and the appropriate filter selected can be seen below.



Pre-Processing

The procedure below was used as a preliminary means for determining features (heuristics) that were unrelated to the final classification and should, therefore, be eliminated.

1. On the drop down list on top of the graph, the class “Winner (nom)” was selected. This colors the bars in the graphs based on our result of interest which is if the given board eventually led the game to a win or a lose.
2. The desired feature to be inspected was selected with a single click of the mouse on the large menu on the left.
3. The graph was then inspected to determine the details including the following:
    a. Are there multiple bars? If not, eliminate feature.
    b. Is the proportion of winner = 1 to winner = 2 in different bars different? If not, eliminate feature.

After pre-processing a list of features could be eliminated from the data set. These features include subsets of the calcrow feature, and the HasWin feature. The other features were left in the analysis.

Classifier

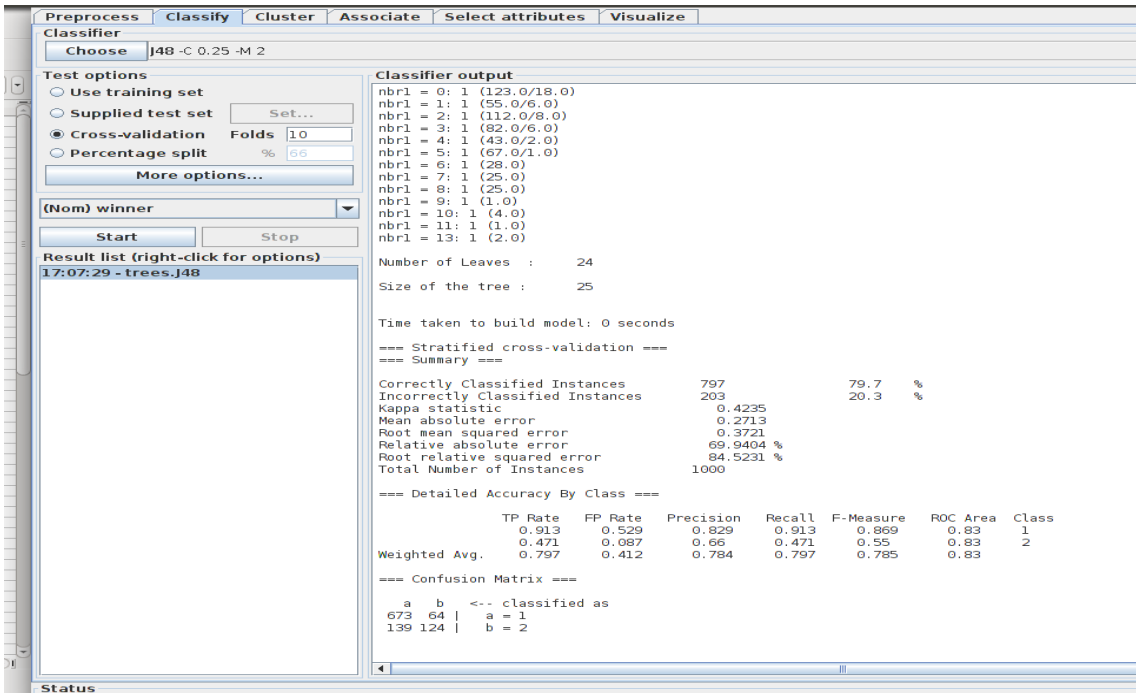
Once the preprocessing phase was complete the next step was to move onto the classifier stage. The process below illustrates the steps required to use the classify function in Weka.

1. Click on the tab that says classify.
2. In the Classifier section below the tabs click the button choose.
3. Once choose has been selected click on the folder that says trees.
4. Scroll down and then choose J48.
5. In test options select Cross-Validation with 10 Folds

The steps mentioned above will setup the program. In order to measure the accuracy of one feature at a time use the following process.

1. Return to the preprocess tab.

2. In the Attributes section click on the All button.
3. Then uncheck the attributes that are going to be measured. For instance “Winner” and “nbr1”.
4. Once those attributes have been selected remove the rest, by clicking the remove button.
5. Return to the classify tab, and on the drop down menu to the left choose (Nom) winner.
6. Then press the start button.



The figure above displays the view of the Classify section, the data of interest is classified under stratified cross-variation.

The classify tab in the weka UI is where the user can train and test learning schemes that classify or perform regression. The choice of a classifier was given in class as J48 (J48 is a clone of the C4.5 decision tree learner). One measure of the performance of a classifier is by counting the proportion of correctly predicted examples in an unseen test dataset. This value is called the accuracy. The simplest method uses a training set and a test set which are mutually independent. This practice is referred to as a hold-out estimate. A more elaborate method is called cross-validation.

Cross-Validation and Decision Tree Results

Background

In this section four common methods for validation of a decision tree will be presented. This includes: Leave One Out Cross-Validation; K-Fold Cross Validation; Cumulative Cross Validation; and, Percentage Split. A rationale will then be provided as to why K-Fold Cross Validation was chosen for the analysis.

The first validation technique is called “Leave one out cross-validation (loo-cv)”, loo-cv selects m = N training sets simply by taking the data set D and removing the ith record for training set D_i^t . The validation set consist of just the ith single record. Loo-cv does not always produce accurate performance estimates.
The cross-validation algorithm is “K-fold cross-validation (k-fold cv)” which splits the data D in m approximately equal parts $D_1,...,D_M$. Training set D_i^t can be obtained by removing part D_i from D. Typical values for m are 5, 10 and 20. With m = N, k-fold cross-validation becomes loo-cv.
The third type of validation is called “Cumulative cross-validation (cumulative cv)” which starts with an empty data set and adds instances item by item from D. After each time an item is added the next item to be added is classified using the then current state of the Bayes network.
The final validation method discussed is referred to as Percentage Split in WEKA. In this modality instead of splitting the data into equal segments such as is the case in K-Fold Validation, the data is split into two segments of unequal size determined by the percentage.
The cross validation scheme that is being used is K-fold CV. There are two main reasons for this choice. Firstly, K-fold cross validation is provided in the classify tab in WEKA whereas loo-cv and cumulative validation are not. Second, K-Fold CV reduces the chances of overfitting when compared to a simple split. The reason for that is that in the percentage split, only two domains are created. Since the training domain is, usually, significantly larger than the test domain, this leads to fitting of the model to a large number of data points and an attempt for testing for generability in a small number (unrepresentative) of the whole sample. Notice that WEKA uses a variant of K-Fold CV referred to as stratified CV. Which uses of slightly modified folds to create the same class distributions in each fold as in the complete dataset. Stratified cross validation will be the primary source of data that will be used to analyze the performance of a feature.

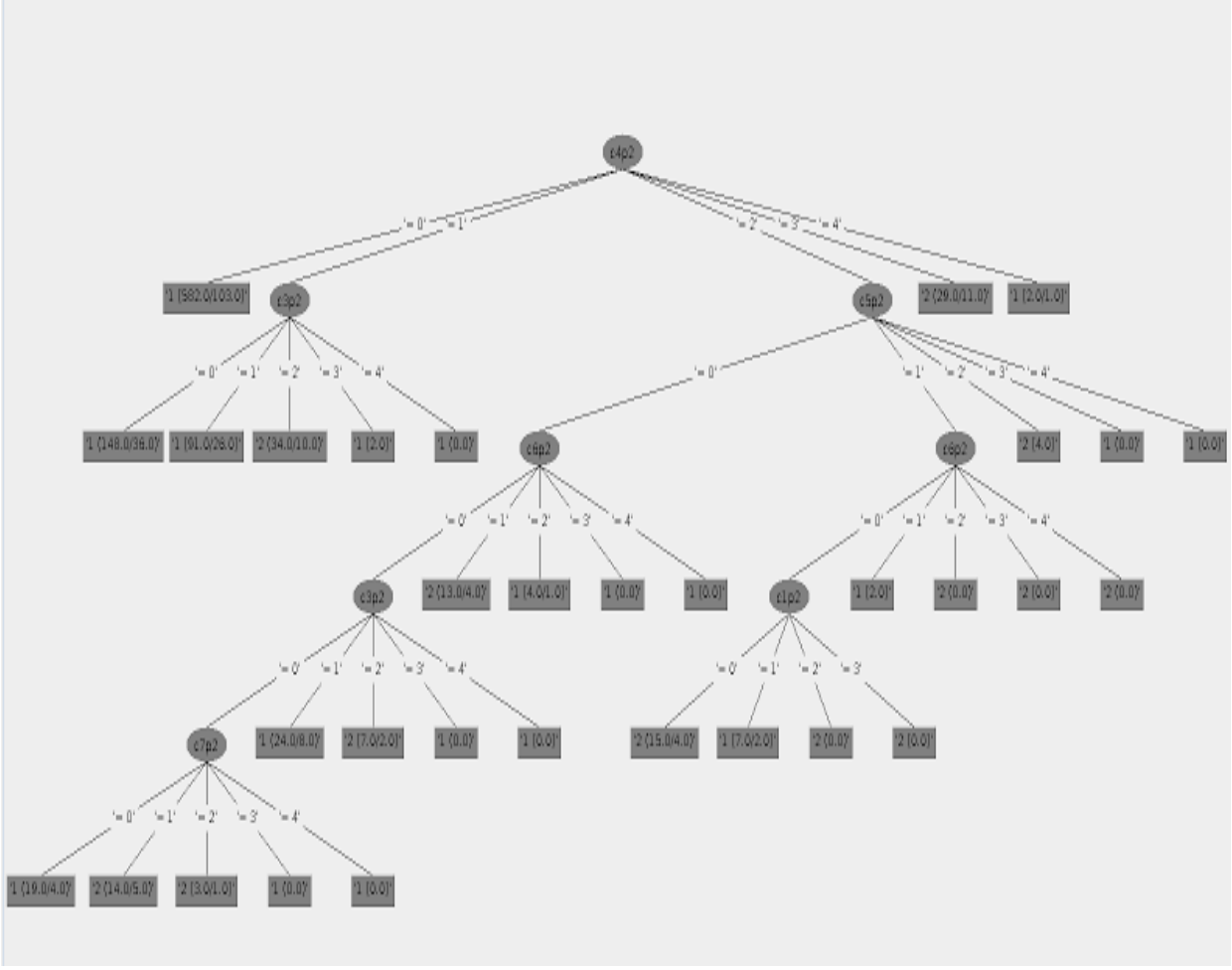
Using Cross-Validation in WEKA (Results)

The performance for the different attributes is listed below. The metrics being used to analyze the heuristics are: the percentage of correctly classified instances, the percentage of incorrectly classified instances, and the kappa statistic. The kappa statistic measures the agreement of prediction with the true class (ending state) 1.0 signifies complete agreement. Notice that all values are derived using K = 10.

Attribute	Correct Classification	Incorrect Classification	Kappa Statistic
r4p1 through r6p1	73.2%	26.8%	0.0113
r4p2 through r6p2	73.7%	26.3%	0
CalcCols player 1	73.7%	26.3%	0
CalcCols player2	75.9 %	24.1%	0.251

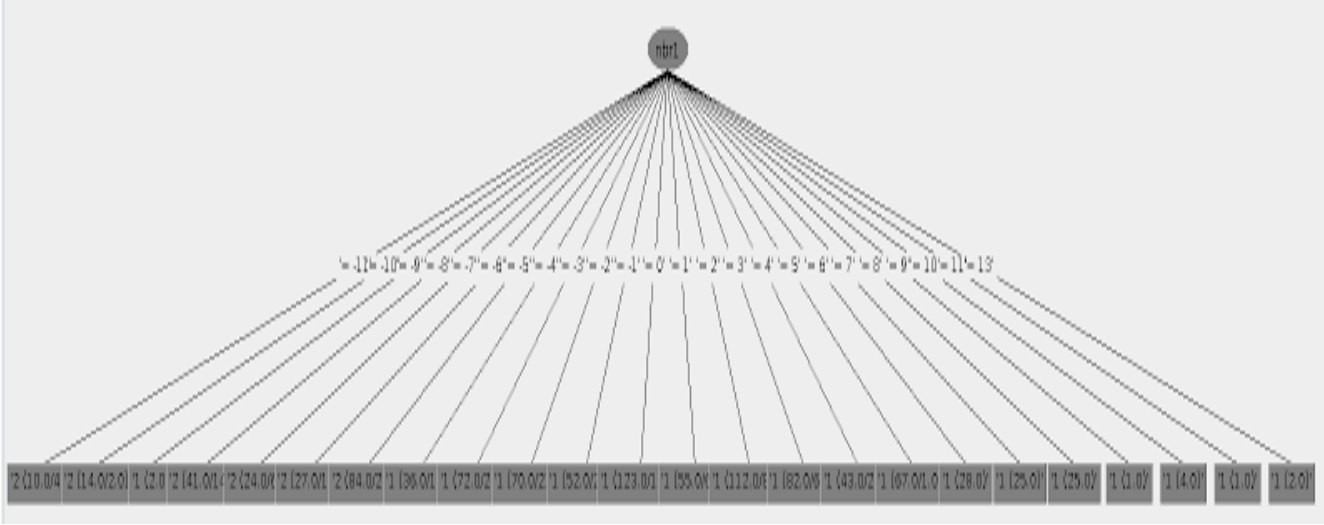
nbr1	79.7%	20.3%	0.4235
nbr2	80.4%	19.6%	0.4224
nbr3	73.7 %	26.3 %	0
nb4	73.7%	26.3%	0
nc2p1 & nc2p2	73.7%	26.3%	0
nc3p1 & nc3p2	73.7%	26.3%	0

For all the features shown, the only ones that seem to correctly predict the true outcome of the game are the classes nbr1, nbr2, and CalcCols for player 2. Taking a closer look at the decision trees for each of these three features it can be seen how each of them leads to a win state.

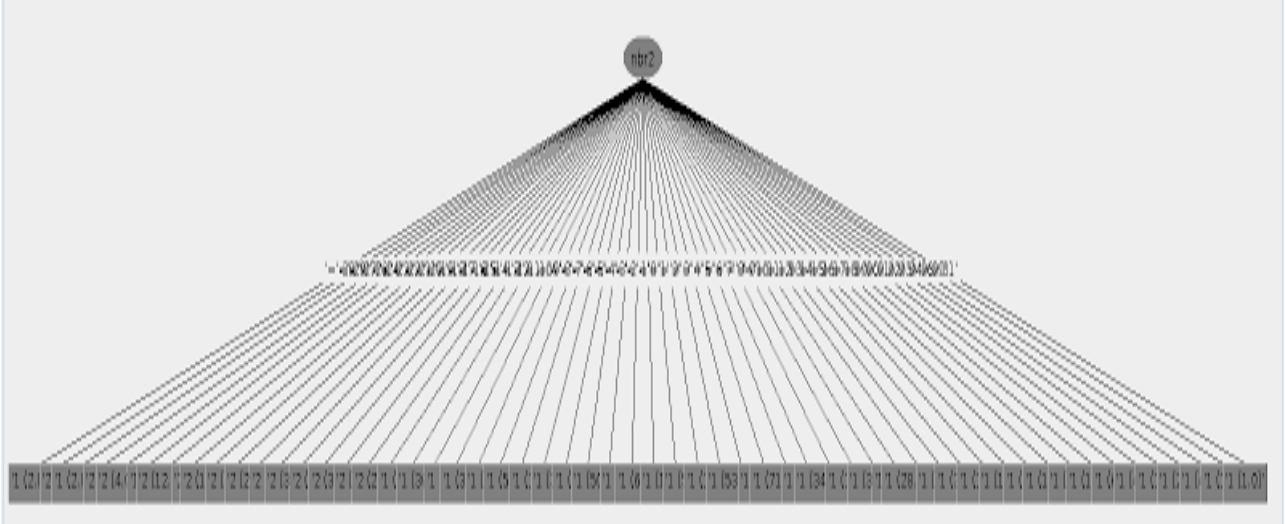


Above is the decision tree for CalcCol for player 2. As can be observed, the decision tree starts by checking whether or not player 2 has played in column 4 (the middle column).

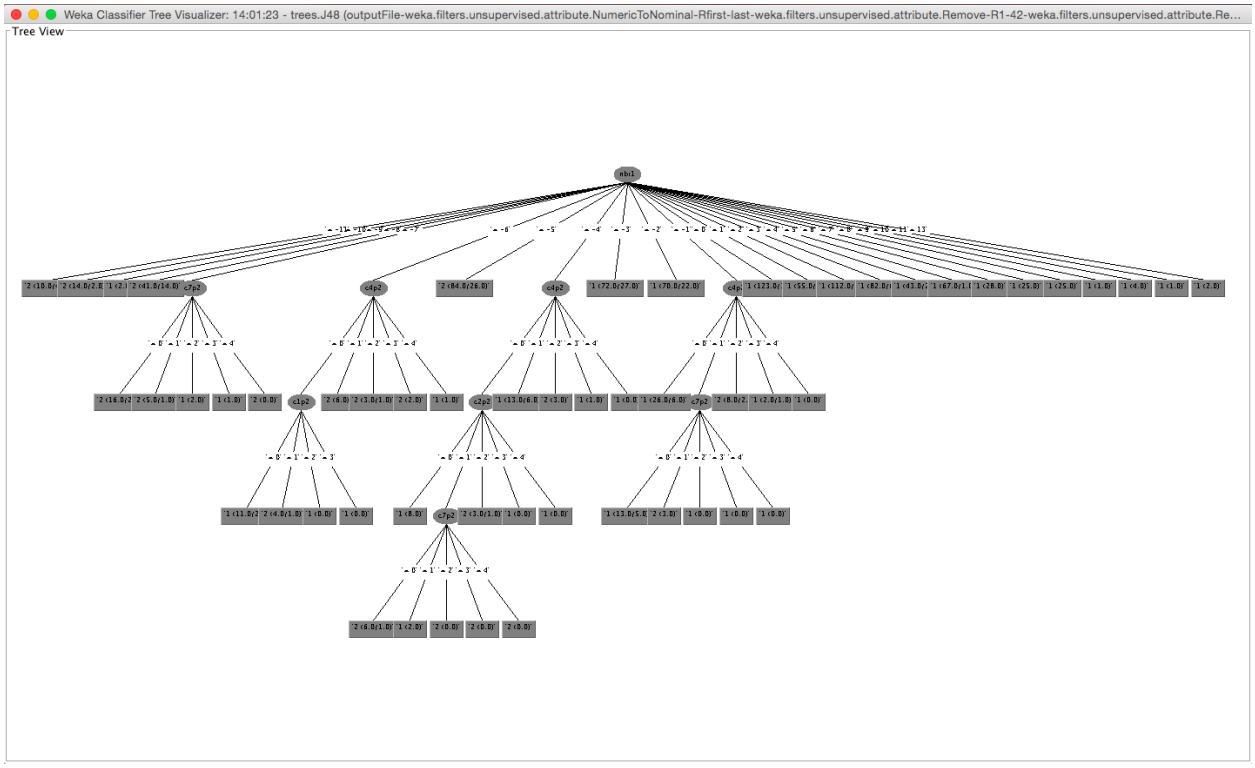
Depending on the number of pieces in column four the tree decides between its two major branches. Ex: If player 2 possesses less than 2 pieces on the center column the tree branches to the left.



The decision tree for the neighbor metric with radius 1 (nbr1) is shown above and the decision tree for the neighbor metric with radius 2 is shown below. Notice that in both cases there is an agreement between the expected results from the preprocessing and the selection from the tree. The result being, that high values of this heuristic tend to imply a better game state for player 1.



Shown below is the tree for the combination of the best three heuristics: CalcCols, NBr1 and NBr2. As expected, NBr1 becomes the root node. Depending on its value, the other heuristics come in to form final decisions.


Weka Classifier Tree Visualizer: 14:01:23 - trees.J48 (outputFile-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.unsupervised.attribute.Remove-R1-42-weka.filters.unsupervised.attribute.Re...
Tree View