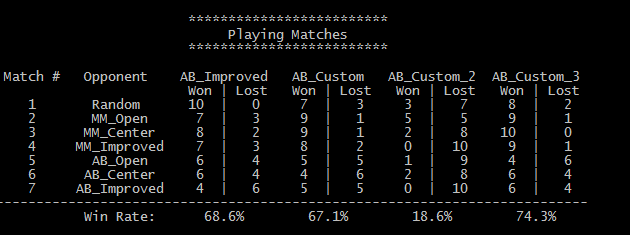
In my program, the evaluations functions were as follows:

* **custom\_score\_3**: this evaluation function evaluates the board based on the number of legal moves available for the computer player and also considers distance away from the center.
* **custom\_score\_2**: this evaluation function is an enhanced version of the of the custom\_score\_3. #\_my\_moves - 2 \* #\_opponent\_moves
* **custom\_score**: This function evaluates based three factors: number of moves for both players and distance from the center. The function evaluates the position more positively if the player’s distance is closer to the center of the board.

The result is shown below:



custom\_score\_3 The logic behind this function makes sense because the more moves available to a given player that should be positive.  This function performed the best among all other functions. custom\_score\_2’s logic is similar to custom\_score3 but it gives more weight if a move creates less available moves eliminating. Overall custom\_score\_2 gave a poor result. The reasoning may be that in trying to eliminate the opponent’s options, the agent may not be choosing more optimal moves. custom\_score’s logic is that the closer to the center of the board, the more likely a player would have more options down the road.

Overall I had interesting and mixed results.  The agent had overall good results but did not perform well against AB\_Improved or AB\_Open. For some games, a simple custom\_3 function performed better than rest of the evaluation functions. Perhaps larger game plays may provide a better picture of the performance among the evaluation functions.

Based on the above data I would recommend the function custom\_score3, which evaluates the position based on the difference between the number of agent’s moves and opponent’s moves. (1) It’s easy to understand and has an intuitive logic behind it; (2) it had the best performance compared with other functions; (3) and it considers two factors when evaluating positions (although one would need to normalize when combining multiple heuristics).