My code

Custom score

```
def custom_score(game, player):
   if game.is_winner(player):
        return float("inf")
    if game.is_loser(player):
        return float("-inf")

    my_moves = len(game.get_legal_moves(player))
    opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float(my_moves - math.exp(opponent_moves))
```

Custom Score 2

```
def custom_score_2(game, player):
       if game.is_winner(player):
        return float("inf")
    if game.is_loser(player):
        return float("-inf")

    my_moves = len(game.get_legal_moves(player))
    opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float(my_moves - (2 * opponent_moves))
```

Customer score 3

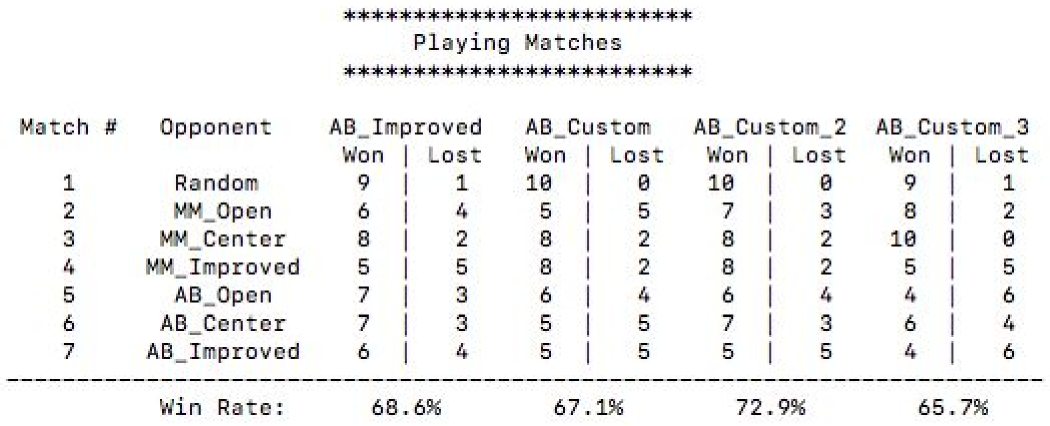
```
def custom_score_3(game, player):
     if game.is_winner(player):
        return float("inf")
    if game.is_loser(player):
        return float("-inf")

    opp_location = game.get_player_location(game.get_opponent(player))
    if opp_location == None:
        return 0
    my_locatation = game.get_player_location(player)
    if my_locatation == None:
        return 0

    return float(abs(sum(opp_location) - sum(my_locatation)))
```

Custom_score is focused on finding a path where my agent has more moves than the exponential of the number of moves that the opponent will play. Custom_score_2 finds a path where my agent has twice the moves available to the opponent at the same node. It is aggressive play that chases after the opponent.

Custom_score_3 gives the maximum distance between my agent and the opponent. It gives the absolute difference between the sum of the location vectors.



```
                                *************************
                                     Playing Matches
                                *************************

 Match #     Opponent     AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                           Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1         Random        9  |   1    10  |   0    10  |   0     9  |   1
    2         MM_Open       6  |   4     5  |   5     7  |   3     8  |   2
    3        MM_Center      8  |   2     8  |   2     8  |   2    10  |   0
    4       MM_Improved     5  |   5     8  |   2     8  |   2     5  |   5
    5         AB_Open       7  |   3     6  |   4     6  |   4     4  |   6
    6        AB_Center      7  |   3     5  |   5     7  |   3     6  |   4
    7       AB_Improved     6  |   4     5  |   5     5  |   5     4  |   6
--------------------------------------------------------------------------
            Win Rate:      68.6%        67.1%        72.9%        65.7%
```

The opponent agent that randomly chooses its move on each turn performs the worst against my agent.

In the grand scheme of things the performance of the opponent agents are in the following order. The AlphaBeta agents performed the best with the improved_score heuristic having the best performance. The Minimax agent comes next with impoved_score as its best heuristic and the Random agent last.This is probably because the Alphabeta agent uses iterative deepening which goes deeper than the Minimax agent that is depth limited.

I would recommend custom_score_2 because the following reasons:

1. **Winning rate: it has the best overall winning rate**
2. **Complexity: This is quite simple and straightforward.**
3. **Depth:It deals better with horizon effect, since it looks a node further than the set node since it considers the opponent.**