Analysis of Evaluation Functions:

performance of custom_score's functions:-

```
                                                          *************************
                                                               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        9  |  1      9  |  1      9  |  1
    2         MM_Open      9  |  1        9  |  1      8  |  2      7  |  3
    3        MM_Center     9  |  1        9  |  1      7  |  3      7  |  3
    4       MM_Improved    8  |  2        6  |  4      7  |  3      6  |  4
    5         AB_Open      6  |  4        5  |  5      4  |  6      6  |  4
    6        AB_Center     5  |  5        5  |  5      4  |  6      3  |  7
    7       AB_Improved    4  |  6        2  |  8      1  |  9      5  |  5
--------------------------------------------------------------------------
           Win Rate:      71.4%         64.3%        57.1%        61.4%
```

AB_Custom:-

Opponent	Won | Lost
Random	9 | 1
MM_Open	9 | 1
MM_Center	9 | 1
MM_Improved	6 | 4
AB_Open	5 | 5
AB_Center	5 | 5
AB_Improved	2 | 8

Win Rate: 64.3%
In this we have defined the function as below

def custom_score(game, player):

```
    ply = game.get_legal_moves(player)

    opp = game.get_legal_moves(game.get_opponent(player))

    if len(ply) == 0:
        return float('-inf')

    else:

        data = (len(ply)-len(opp))*(len(ply)-len(opp))

        second_data = custom_score_2(game, player)

        second_data = second_data*second_data

        if len(ply) > len(opp):

            return float(data+second_data)

        else:

            return float(second_data)
```

As we can see in this above heuristic function I am giving attention to two factors. One is the current player's available move subtracted by opponent's available move. And I am adding this results square root to the result of second custom score result' square root. those both combined gives a good and reasonable heuristic function.

AB_custom_2:-

In this we have defined the function as below.

```
def custom_score_2(game, player):
      a = set(game.get_legal_moves(player))
      b = set(game.get_legal_moves(game.get_opponent(player)))
      return float(len(list(a-b)))
```

In this custom score function, I am only subtracting those moves from player catalog which are also available for opponent's legal moves. It is good to check that how many places are solely available for our agent.

AB_custom_3:-

In this we have defined the function as below.

```
def custom_score_3(game, player):
    ply_move = set(game.get_legal_moves(player))
    opp_move=set(game.get_legal_moves(game.get_opponent(player))
    solo_ply_move=list(player_move-opponent_move)
    v=0
    for x in solo_player_move:
        v = max(v, len(game.forecast_move(x).get_legal_moves(player)))
    return float(v+len(ply_move))
```

In this function we are going one more depth and checking only those which are solely available to our player. and returning the maximum available moves in next level with addition of current available moves.

Best Heuristic Function:-

After analysing all the above heuristic function, custom_score is best suited for my agent except AB_Improved which also has a good heuristic function. Reasons to choose this function:-
1.It uses current player available move, If our agent’s available move is more than opponent’s available move than it gives us good score. and The score is better when the margin of player’s moves and opponent’s moves is bigger(player’s move must be more than opponent’s move).
2. And with the help of custom_score_2 we can make this evaluation function strong by adding custom_score_2’s value to this custom_score method, I get available moves as set of available moves for both agent and opponent player. And get only solely available move for agent. If solely available move for our agent is more than our heuristic value becomes strong.
3. If agent’s move subtract by opponent’s move gives negative value than we only return custom_score_2’s value because it will be beneficial for agent that it has some solo move that it can play.