Heuristic Analysis

The goal of the `custom_score()` function is to estimate the score of a terminal (at a given depth) node in the game tree search. I implemented the following versions of the `custom_score()` function.

custom_score_1()

This version is very similar to the improved_score function in the `sample_agent.py`. Instead of returning the difference between the legal moves of the active and the inactive players it returns the following

```
score = math.exp(len(active_moves) - len(inactive_moves))
```

The goal is to amplify the difference in the number of legal moves available to the two players, so that the scores propagated up the tree are easier to distinguish. Here are the comparative scores for this function.

```
*************************
 Evaluating: ID_Improved
*************************
score func  improved_score

Playing Matches:
----------
  Match 1: ID_Improved vs   Random    Result: 17 to 3
  Match 2: ID_Improved vs   MM_Null   Result: 17 to 3
  Match 3: ID_Improved vs   MM_Open   Result: 15 to 5
  Match 4: ID_Improved vs MM_Improved     Result: 8 to 12
  Match 5: ID_Improved vs   AB_Null   Result: 16 to 4
  Match 6: ID_Improved vs   AB_Open   Result: 14 to 6
  Match 7: ID_Improved vs AB_Improved     Result: 12 to 8

Results:
----------
ID_Improved          70.71%

*************************
   Evaluating: Student
*************************
Playing Matches:
```

```
----------
  Match 1:   Student   vs   Random    Result: 16 to 4
  Match 2:   Student   vs   MM_Null   Result: 16 to 4
  Match 3:   Student   vs   MM_Open   Result: 12 to 8
  Match 4:   Student   vs MM_Improved     Result: 16 to 4
  Match 5:   Student   vs   AB_Null   Result: 18 to 2
  Match 6:   Student   vs   AB_Open   Result: 12 to 8
  Match 7:   Student   vs AB_Improved     Result: 12 to 8

Results:
----------
Student               72.86%
```

custom_score_2

In isolation if the two players get isolated from each other the one with more moves always wins. In this function I try to detect if the two players are isolated. I find the bounding boxes of the legal moves for the two players and calculate the overlap between the two boxes. If there is no overlap I return a high score, otherwise I return the difference in the legal moves. Of course this is not a perfect estimate as the isolation might be broken at a deeper level, but it looks worth a try.

```
*************************
 Evaluating: ID_Improved
*************************
score func  improved_score

Playing Matches:
----------
  Match 1: ID_Improved vs   Random    Result: 14 to 6
  Match 2: ID_Improved vs   MM_Null   Result: 18 to 2
  Match 3: ID_Improved vs   MM_Open   Result: 13 to 7
  Match 4: ID_Improved vs MM_Improved     Result: 13 to 7
  Match 5: ID_Improved vs   AB_Null   Result: 18 to 2
  Match 6: ID_Improved vs   AB_Open   Result: 13 to 7
  Match 7: ID_Improved vs AB_Improved     Result: 10 to 10

Results:
----------
ID_Improved           70.71%

*************************
   Evaluating: Student
```

```
*************************
Playing Matches:
----------
  Match 1:   Student   vs   Random    Result: 17 to 3
  Match 2:   Student   vs   MM_Null   Result: 18 to 2
  Match 3:   Student   vs   MM_Open   Result: 13 to 7
  Match 4:   Student   vs MM_Improved     Result: 13 to 7
  Match 5:   Student   vs   AB_Null   Result: 17 to 3
  Match 6:   Student   vs   AB_Open   Result: 9 to 11
  Match 7:   Student   vs AB_Improved     Result: 19 to 1

Results:
----------
Student               75.71%
```

This looks better than the first function.

custom_score_3

My third function is similar to the second one, except that in the case isolation is not detected I return the exponentiated difference in the legal moves, instead of just the difference.

```
*************************
 Evaluating: ID_Improved
*************************
Playing Matches:
----------
  Match 1: ID_Improved vs   Random    Result: 15 to 5
  Match 2: ID_Improved vs   MM_Null   Result: 16 to 4
  Match 3: ID_Improved vs   MM_Open   Result: 14 to 6
  Match 4: ID_Improved vs MM_Improved     Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null   Result: 16 to 4
  Match 6: ID_Improved vs   AB_Open   Result: 12 to 8
  Match 7: ID_Improved vs AB_Improved     Result: 14 to 6

Results:
----------
ID_Improved           69.29%

*************************
   Evaluating: Student
*************************
Playing Matches:
----------
```

```
  Match 1:   Student   vs   Random     Result: 14 to 6
  Match 2:   Student   vs   MM_Null    Result: 19 to 1
  Match 3:   Student   vs   MM_Open    Result: 14 to 6
  Match 4:   Student   vs MM_Improved     Result: 13 to 7
  Match 5:   Student   vs   AB_Null    Result: 20 to 0
  Match 6:   Student   vs   AB_Open    Result: 14 to 6
  Match 7:   Student   vs AB_Improved     Result: 11 to 9

Results:
----------
Student                 75.00%
```

Analysis

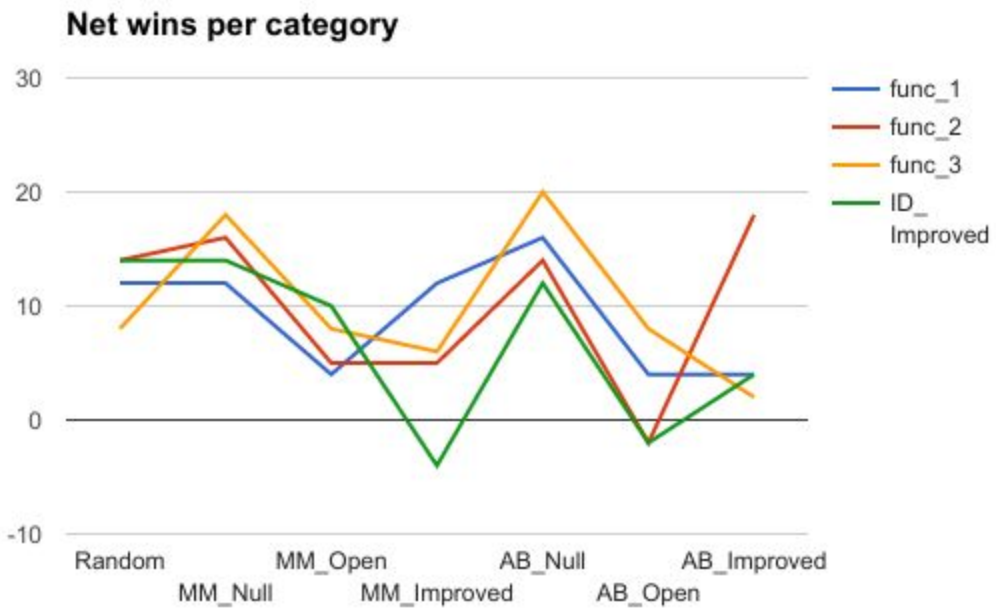

Figure 1: Net wins for each scoring function per category

Figure 1 above shows the net wins (wins - losses) for each scoring function. Let's look at the characteristics of the three functions.

1. `custom_score_3` and `custom_score_2` outscore ID_Improved
2. `custom_score_3` has a slightly less overall score as `custom_func_2()` (75% vs 75.71%), however the performance of this function is more consistent. With

`custom_score_3()` the student player wins in all categories, which is not the case with `custom_score_2()`.

3. While `custom_func_3()` has a more complicated implementation compared to `custom_func_1()` it exploits a key winning strategy in isolation, which is - if the players get isolated the player with more available moves always wins.

Given these three observations I chose `custom_score_3()` for submission.