Possible Approach to March Madness Dataset

Overview

To solve this problem, we will create 2 entitysets with different “views” of the data. We will perform feature engineering on each view and then merge the results together.

The first entityset will create features for a particular pair of teams that are matching. The second entityset will create features for an individual team.

Notes

- This obviously doesn’t follow the actual dataset, but hopefully you can see the parallel’s between my schema and the actual data.
- In your work, you should rename entities and column names to make things more clear

Entity Set 1

matchups entity

This entity has a row for every unique pairing of teams irrespective of home or away

match_id	team 1	team 2
duke_unc	duke	unc
kentucky_louisville	kentucky	louisville

matchup_logs entity

this entity has the outcomes of the matchups that occured

game	match_id	team_1_win	team_2_win	team*1*points	team*2*points
game_id_1	duke_unc	True	False	100	98
game_id_2	duke_unc	False	True	88	108
game_id_3	kentucky_louisville	True	False	86	76
game_id_4	duke_unc	?	?	?	?

Notes:

- The entities are related by the `match_id` column
- The target entity for DFS would be `matchup_logs`
- The last row represents a match up we want to make a prediction for
- I excluded it, but we need to add a time_index column for each game, so we can use cutoff times
- When we call DFS, we pass cutoff times for each instance in matchup_logs where the time is the time of the game. This will make sure we only use data from previous games when calculating features
- When we run DFS, we will want to use cutoff times
- To create feature like "percent of time duke won against unc" add interesting values `[True, False]` to `team_1_win`, `team_2_win` variables

Entity Set 2

team entity

this entity has a row for each team

team
duke
unc
louisville
kentucky

team_game_logs entity

This entity contains the outcomes of games for a single team's point of view. This means each game will have

2 rows for each team

id	game	team	opponent	side	team_points	opponent_points	win
1	game_id_1	duke	unc	Home	100	98	True
2	game_id_1	unc	duke	Away	98	100	False
3	game_id_2	duke	unc	Away	88	108	False
4	game_id_3	unc	duke	Home	108	88	True
5	game_id_3	louisville	kentucky	Home	76	86	False
6	game_id_3	kentucky	louisville	Away	86	76	True
7	game_id_4	duke	unc	Home	?	?	?
8	game_id_4	unc	duke	Away	?	?	?

Notes

- these entities are connected by a relationship between `team` columns
- the target entity will be team game logs
- I excluded it, but we need to add a time_index column for each game, so we can use cutoff times
- When we call DFS, we pass cutoff times for each instance in team*game*logs where the time is the time of the game. This will make sure we only use data from previous games when calculating features
- The value for `game_id_*` should match between entitysets
- To create feature like “average points when home team” add interesting values `['Home', 'Away']` to `side`

Merging to get final matrix

After features are calculated for both entities, we can merge together based on

1. Merge output of ES 1 with output of ES2 on `game`, `team_1` / `game`, `team`
2. Then merge output of step 1 with output of ES2 on `game`, `team_2` / `game`, `team`