**My Approach:**

Well the problem statement may seem easy but it’s not. Every time I felt like “I am so close but yet so far”.

With that’s been said, I have shared difficulty faced, How I tackled that and obviously the room for improvement for this given problem statement below:

**1. Difficulty Faced and How I tried to solve it:**

1. With the given link I was confused what data should be used at first because there’s no scarcity of data if continue clicking inside the links of individual players.

* So, I kept it simple for this, I just copied pasted team players data from every single team (8 times) in an excel file (called it “Raw\_Data.xlsx”)

1. Data pre-processing and data cleansing and feature engineering have been a quite a task in every problem now a then so why not here?!

* With that Raw data, I imported it in my Jupyter notebook and python (mainly pandas) makes thing easier for us. No manual work was required, except for one place where I had to add a feature “IsUncapped” (whether a player is uncapped or not) and it was not readily available in the link, so here my cricketing knowledge came into play and I hardcoded that feature (Wasn’t hectic at all) and saved into a clean excel file “Final\_df.xlsx”.

1. Now I had the cleaned data, still was not getting a clue on selecting the team given the constraints on each team (Each team have 4 batsmen, 2 allrounders, 1 wicketkeeper and 4 bowlers and Teams have 4 overseas players and atleast 1 uncapped player).

* I didn’t have enough data to train a model on, so build my own model.
* First approach was “why don’t I randomly pick 11 players from the selected team and set a constraint using if-ese condition choose that configuration? Well as it may seem an idea but then I thought let’s see the worst-case time complexity for selecting 11 members from 25 member i.e.

**25C11 = 4,457,400,** even for random selection it was higher, so this approach went for a toss because as I even tried, it took very long time.

* Then I extended that approach and I grouped the feature “Playing\_role” together for required constraints and got the indices of them.

e.g. for a particular 25 members now, I had segregated number of different playing roles members:

Batsman & Captain = 1, Batsman = 8, Wicketkeeper = 2, Allrounders = 5, Bowlers = 9

Now I had to randomly pick only: **1C1 + 8C3 + 2C1 + 5C2 + 9C4** combinations at my worst case after imposing the secondary constraints like 4 overseas player and atleast 1 uncapped player. This model started working fine.

1. When everything looks fine then there’s a problem. When I generalised this model, for teams like “Kings X1 Punjab”, “Kolkata Knight Riders”, “Sunrisers Hyderabad” still it was not converging as I set my constrains of only “1 uncapped player” in my model initially.

* I was not sure why it was failing at first then I plotted some histograms (in “**EDA\_Team11.ipynb**”) and saw that No. of Uncapped players specially these teams were higher, like more than 12 that’s why it was not satisfying the constraints, so I cut it some slack and as I increased no. of uncapped players constraint, it again started performing well and it was giving fairly good team combinations satisfying all my constraints.

1. I always wanted to serve this end to end, but I was not expert in this.

* I had little knowledge in Flask and Django so used Django here as I had very little time and creating a WebApp in a Day wasn’t easy. So, I tried my level best to serve the purpose.

**2. Advantages of the Model:**

1. This model works fairly well in terms of constraints imposed

**3. Drawbacks of the Model:**

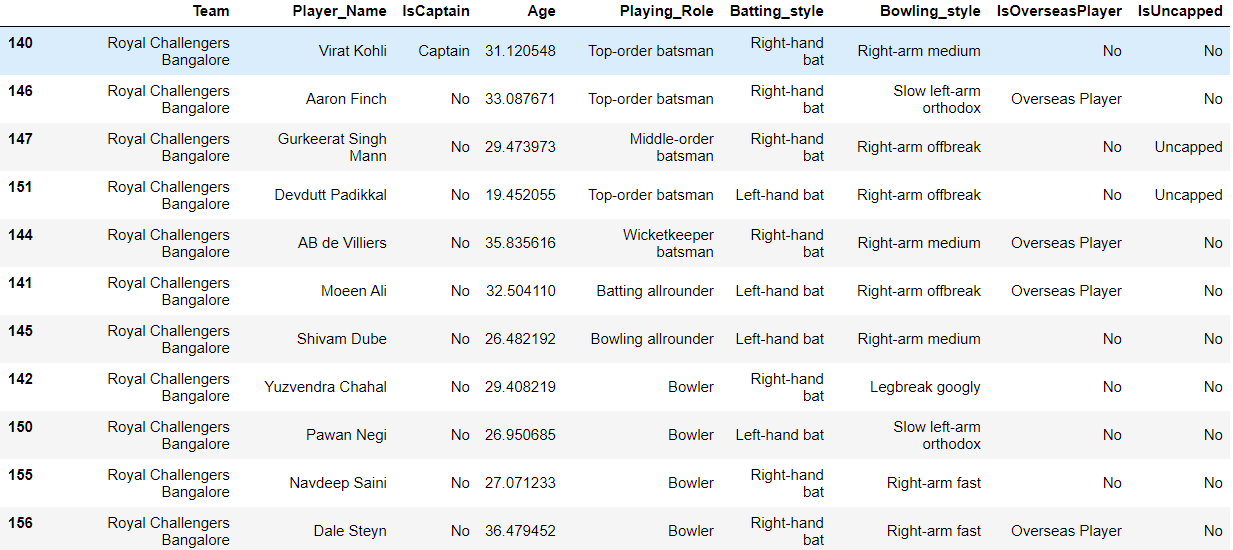
1. The team combination keeps changing each time I select the team as this model works on random search given the constraints.
2. Model does not include stats of previously played matches data.

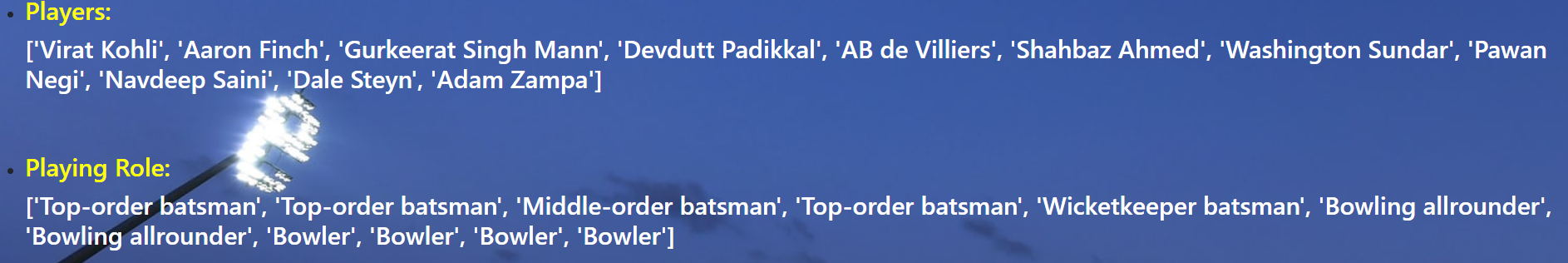
**4. Room for Improvements:**

1. Due to time constraints, records of previously matched played stats could not be added because if I had this, it could become a simple problem. I would have given more “weights” to those players who have played previous 2 or 3 matches to further reduce my selection procedure.
2. WebApp can be improved by a huge margin as I tried with little or no previous knowledge in Flask or Django.

One could find many of loop holes with the deployment architectural pattern.

Just wanted to Showcase how well the model depicts the playing 11 (Jupyter notebook and webapp view respectively):





Thanks,

Saurabh Kumar