Name: Tom Adoni

Date: 8/27/2023

Introduction

The purpose of this project is to gauge your technical skills and problem solving ability by working through something similar to a real NBA data science project. You will work your way through this jupyter notebook, answering questions as you go along. Please begin by adding your name to the top markdown chunk in this document. When you're finished with the document, come back and type your answers into the answer key at the top. Please leave all your work below and have your answers where indicated below as well. Please note that we will be reviewing your code so make it clear, concise and avoid long printouts. Feel free to add in as many new code chunks as you'd like.

Remember that we will be grading the quality of your code and visuals alongside the correctness of your answers. Please try to use packages like pandas/numpy and matplotlib/seaborn as much as possible (instead of base python data manipulations and explicit loops.)

WARNING: Your project will **ONLY** be graded if it's knit to an HTML document where we can see your code. Be careful to make sure that any long lines of code appropriately visibly wrap around visibly to the next line, as code that's cut off from the side of the document cannot be graded.

Note:

Throughout this document, any `season` column represents the year each season started. For example, the 2015-16 season will be in the dataset as 2015. For most of the rest of the project, we will refer to a season by just this number (e.g. 2015) instead of the full text (e.g. 2015-16).

Answers

Part 1

Question 1:

- 1st Team: 20.1 points per game
- 2nd Team: 19.3 points per game
- 3rd Team: 17.6 points per game
- All-Star: 17.0 points per game

Question 2: 3.68 Years

Question 3:

- Elite: 2 players.
- All-Star: 1 players.
- Starter: 8 players.
- Rotation: 9 players.
- Roster: 10 players.
- Out of League: 43 players.

Open Ended Modeling Question: Please show your work and leave all responses below in the document.

Part 2

Question 1: 28.9%
Question 2: Written question, put answer below in the document.
Question 3: Written question, put answer below in the document.

Setup and Data

In [1]:
```python
import pandas as pd
awards = pd.read_csv("awards_data.csv")
player_data = pd.read_csv("player_stats.csv")
team_data = pd.read_csv("team_stats.csv")
rebounding_data = pd.read_csv("team_rebounding_data_22.csv")
```

Part 1 -- Awards

In this section, you're going to work with data relating to player awards and statistics. You'll start with some data manipulation questions and work towards building a model to predict broad levels of career success.

Question 1

QUESTION: What is the average number of points per game for players in the 2007-2021 seasons who won All NBA First, Second, and Third teams (**not** the All Defensive Teams), as well as for players who were in the All-Star Game (**not** the rookie all-star game)?

In [2]:
```python
first_team = awards.loc[awards['All NBA First Team'] == 1]

first_team_ids = first_team['nbapersonid'].tolist()

first_team_data = player_data[player_data.nbapersonid.isin(first_team_ids)]

first_num_games = first_team_data['games'].sum()
first_num_points = first_team_data['points'].sum()

first_ppg = first_num_points / first_num_games
print(first_ppg)
```
```
20.13130705734434
```

In [3]:
```python
second_team = awards.loc[awards['All NBA Second Team'] == 1]

second_team_ids = second_team['nbapersonid'].tolist()

second_team_data = player_data[player_data.nbapersonid.isin(second_team_ids)]

second_num_games = second_team_data['games'].sum()
second_num_points = second_team_data['points'].sum()

second_ppg = second_num_points / second_num_games
print(second_ppg)
```
```
19.300035832304815
```

In [4]:
```python
third_team = awards.loc[awards['All NBA Third Team'] == 1]

third_team_ids = third_team['nbapersonid'].tolist()

third_team_data = player_data[player_data.nbapersonid.isin(third_team_ids)]

third_num_games = third_team_data['games'].sum()
third_num_points = third_team_data['points'].sum()

third_ppg = third_num_points / third_num_games
print(third_ppg)
```
```
17.62783493434889
```

In [5]:
```python
allstar_team = awards.loc[awards['all_star_game'] == True]

allstar_team_ids = allstar_team['nbapersonid'].tolist()

allstar_team_data = player_data[player_data.nbapersonid.isin(allstar_team_ids)]

allstar_num_games = allstar_team_data['games'].sum()
allstar_num_points = allstar_team_data['points'].sum()

allstar_ppg = allstar_num_points / allstar_num_games
print(allstar_ppg)
```
```
17.002149631802933
```

ANSWER 1:

1st Team: 20.1 points per game
2nd Team: 19.3 points per game
3rd Team: 17.6 points per game
All-Star: 17.0 points per game

Question 2

QUESTION: What was the average number of years of experience in the league it takes for players to make their first All NBA Selection (1st, 2nd, or 3rd team)? Please limit your sample to players drafted in 2007 or later who did eventually go on to win at least one All NBA selection. For example:

- Luka Doncic is in the dataset as 2 years. He was drafted in 2018 and won his first All NBA award in 2019 (which was his second season).
- LeBron James is not in this dataset, as he was drafted prior to 2007.
- Lu Dort is not in this dataset, as he has not received any All NBA honors.

In [6]:
```python
post_2007_data = player_data[player_data['draftyear'] >= 2007]

cols = ["All NBA First Team", "All NBA Second Team", "All NBA Third Team"]
awards['all_nba_team'] = awards[cols].sum(axis = 1)

player_data_draftyear = pd.merge(awards , post_2007_data[['nbapersonid' , 'draftyear']], on = 'nbapersonid')

first_all_nba_team = player_data_draftyear.groupby(['nbapersonid' , 'all_nba_team']).season.min().reset_index()

first_all_nba_team = first_all_nba_team[first_all_nba_team['all_nba_team'] == 1]

draft_year_dict = post_2007_data.set_index("nbapersonid").draftyear.to_dict()

first_all_nba_team['draftyear'] = first_all_nba_team.nbapersonid.map(draft_year_dict)

season_diff = first_all_nba_team['season'] - first_all_nba_team['draftyear']

print(season_diff.mean())
```
```
3.682926829268293
```

ANSWER 2:

3.68 Years

Data Cleaning Interlude

You're going to work to create a dataset with a "career outcome" for each player, representing the highest level of success that the player achieved for **at least two** seasons *after his first four seasons in the league* (examples to follow below!). To do this, you'll start with single season level outcomes. On a single season level, the outcomes are:

- Elite: A player is "Elite" in a season if he won any All NBA award (1st, 2nd, or 3rd team), MVP, or DPOY in that season.
- All-Star: A player is "All-Star" in a season if he was selected to be an All-Star that season.
- Starter: A player is a "Starter" in a season if he started in at least 41 games in the season OR if he played at least 2000 minutes in the season.
- Rotation: A player is a "Rotation" player in a season if he played at least 1000 minutes in the season.
- Roster: A player is a "Roster" player in a season if he played at least 1 minute for an NBA team but did not meet any of the above criteria.
- Out of the League: A player is "Out of the League" if he is not in the NBA in that season.

We need to make an adjustment for determining Starter/Rotation qualifications for a few seasons that didn't have 82 games per team. Assume that there were 66 possible games in the 2011 lockout season and 72 possible games in each of the 2019 and 2020 seasons that were shortened due to covid. Specifically, if a player played 900 minutes in 2011, he **would** meet the rotation criteria because his final minutes would be considered to be 900 *(82/66) = 1118. Please use this math for both minutes and games started, so a player who started 38 games in 2019 or 2020 would be considered to have started 38* (82/72) = 43 games, and thus would qualify for starting 41. Any answers should be calculated assuming you round the multiplied values to the nearest whole number.

Note that on a season level, a player's outcome is the highest level of success he qualifies for in that season. Thus, since Shai Gilgeous-Alexander was both All-NBA 1st team and an All-Star last year, he would be considered to be "Elite" for the 2022 season, but would still qualify for a career outcome of All-Star if in the rest of his career he made one more All-Star game but no more All-NBA teams. Note this is a hypothetical, and Shai has not yet played enough to have a career outcome.

Examples:

- A player who enters the league as a rookie and has season outcomes of Roster (1), Rotation (2), Rotation (3), Roster (4), Roster (5), Out of the League (6+) would be considered "Out of the League," because after his first four seasons, he only has a single Roster year, which does not qualify him for any success outcome.
- A player who enters the league as a rookie and has season outcomes of Roster (1), Rotation (2), Starter (3), Starter (4), Starter (5), Starter (6), All-Star (7), Elite (8), Starter (9) would be considered "All-Star," because he had at least two seasons after his first four at all-star level of production or higher.
- A player who enters the league as a rookie and has season outcomes of Roster (1), Rotation (2), Starter (3), Starter (4), Starter (5), Starter (6), Rotation (7), Rotation (8), Roster (9) would be considered a "Starter" because he has two seasons after his first four at a starter level of production.

Question 3

QUESTION: There are 73 players in the `player_data` dataset who have 2010 listed as their draft year. How many of those players have a **career** outcome in each of the 6 buckets?

In [7]:
```python
players2010 = player_data[player_data.draftyear == 2010]
players2010.nbapersonid.nunique()
```
Out[7]:
```
73
```

In [8]:
```python
awards['MVP'] = awards['Most Valuable Player_rk'] == 1
awards['DPOY'] = awards['Defensive Player Of The Year_rk'] == 1

cols = ["All NBA First Team", "All NBA Second Team", "All NBA Third Team",
        "MVP", "DPOY"]

awards['Elite'] = awards[cols].sum(axis = 1)

awards_draft_year = pd.merge(awards, players2010, on = ["nbapersonid", "season"])

awards_draft_year = awards_draft_year[awards_draft_year.season > 2013]

elite_frequency = awards_draft_year.groupby("nbapersonid").Elite.value_counts().unstack().fillna(0)
```

In [9]:
```python
elite_players = elite_frequency[elite_frequency[1.0]>=2].index
elite_players.shape[0]
```
Out[9]:
```
2
```

In [10]:
```python
all_stars = awards_draft_year[awards_draft_year["all_star_game"].notnull()]
```

In [11]:
```python
allstar_players = (all_stars.groupby("nbapersonid").all_star_game.value_counts().unstack().drop(elite_players)>1).sum()
allstar_players.shape[0]
```
Out[11]:
```
1
```

In [12]:
```python
all_star_players = all_stars.groupby("nbapersonid").all_star_game.value_counts().unstack().drop(elite_players)>1
all_star_players = all_star_players[all_star_players.iloc[:, 0]>0].index
all_star_players
```
Out[12]:
```
Float64Index([202322.0], dtype='float64', name='nbapersonid')
```

In [13]:
```python
starters = awards_draft_year[(awards_draft_year["games_start"] >= 41) & (awards_draft_year["mins"] >= 2000)]
```

In [14]:
```python
starter_vc = starters.nbapersonid.value_counts()
starter_vc
```
Out[14]:
```
202331.0    4
202339.0    4
202322.0    3
202355.0    3
202330.0    3
202326.0    2
202329.0    2
202324.0    2
202340.0    1
202357.0    1
Name: nbapersonid, dtype: int64
```

In [15]:
```python
starter_vc[starter_vc>1].shape[0]
```
Out[15]:
```
8
```

In [16]:
```python
starter_players = starter_vc[starter_vc>1].index
starter_players
```
Out[16]:
```
Float64Index([202331.0, 202339.0, 202322.0, 202355.0, 202330.0, 202326.0,
              202329.0, 202324.0],
             dtype='float64')
```

In [17]:
```python
rotation = awards_draft_year[awards_draft_year.mins.between(1000, 2000, inclusive="left" )]
```

In [18]:
```python
rotation_vc = rotation.nbapersonid.value_counts()
rotation_vc
```
Out[18]:
```
202340.0    5
202330.0    4
202355.0    4
202357.0    4
202324.0    4
202334.0    3
202331.0    3
202322.0    3
202397.0    3
202362.0    2
202498.0    2
202335.0    2
202326.0    2
202391.0    2
202323.0    2
202339.0    2
202325.0    1
202329.0    1
202328.0    1
202337.0    1
202344.0    1
Name: nbapersonid, dtype: int64
```

In [19]:
```python
rotation_vc = rotation_vc.drop(starter_players, errors = "ignore").drop(all_star_players,
                                                                     errors = "ignore").drop(elite_players, errors = "ignore"
```

In [20]:
```python
rotation_vc = rotation_vc[rotation_vc > 1]
rotation_vc.shape[0]
```
Out[20]:
```
9
```

In [21]:
```python
rotation_players = rotation_vc.index
```

In [22]:
```python
roster = awards_draft_year[(awards_draft_year["mins"] >= 1) & (awards_draft_year["mins"] < 1000)]
```

In [23]:
```python
roster_vc = roster.nbapersonid.value_counts()
roster_vc = roster_vc.drop(rotation_players).drop(starter_players,
                                                  errors="ignore").drop(all_star_players,
                                                                        errors = "ignore").drop(elite_players,
                                                                                                errors = "ignore"
roster_vc
```
Out[23]:
```
1626246.0    9
202328.0     6
202344.0     3
202325.0     3
202347.0     2
202337.0     2
202332.0     2
202327.0     2
202620.0     1
202338.0     1
Name: nbapersonid, dtype: int64
```

In [24]:
```python
roster_vc.shape[0]
```
Out[24]:
```
10
```

In [25]:
```python
roster_players = roster_vc[roster_vc>1].index
```

ANSWER 3:

Elite: 2 players.
All-Star: 1 players.
Starter: 8 players.
Rotation: 9 players.
Roster: 10 players.
Out of League: 43 players.

Open Ended Modeling Question

In this question, you will work to build a model to predict a player's career outcome based on information up through the first four years of his career.

This question is intentionally left fairly open ended, but here are some notes and specifications.

1. We know modeling questions can take a long time, and that qualified candidates will have different levels of experience with "formal" modeling. Don't be discouraged. It's not our intention to make you spend excessive time here. If you get your model to a good spot but think you could do better by spending a lot more time, you can just write a bit about your ideas for future improvement and leave it there. Further, we're more interested in your thought process and critical thinking than we are in specific modeling techniques. Using smart features is more important than using fancy mathematical machinery, and a successful candidate could use a simple regression approach.

2. You may use any data provided in this project, but please do not bring in any external sources of data. Note that while most of the data provided goes back to 2007, All NBA and All Rookie team voting is only included back to 2011.

3. A player needs to complete three additional seasons after their first four to be considered as having a distinct career outcome for our dataset. Because the dataset in this project ends in 2021, this means that a player would need to have had the chance to play in the '21, '20, and '19 seasons after his first four years, and thus his first four years would have been '18, '17, '16, and '15. **For this reason, limit your training data to players who were drafted in or before the 2015 season.** Karl-Anthony Towns was the #1 pick in that season.

4. Once you build your model, predict on all players who were drafted in 2018-2021 (They have between 1 and 4 seasons of data available and have not yet started accumulating seasons that inform their career outcome).

5. You can predict a single career outcome for each player, but it's better if you can predict the probability that each player falls into each outcome bucket.

6. Include, as part of your answer:
   - A brief written overview of how your model works, targeted towards a decision maker in the front office without a strong statistical background.
   - What you view as the strengths and weaknesses of your model.
   - How you'd address the weaknesses if you had more time and or more data.
   - A matplotlib or plotly visualization highlighting some part of your modeling process, the model itself, or your results.
   - Your predictions for Shai Gilgeous-Alexander, Zion Williamson, James Wiseman, and Josh Giddey.
   - (Bonus!) An html table (for example, see the package `reactable`) containing all predictions for the players drafted in 2019-2021.

Training the data

I will use training data and build a model that will predict player success by measuring his chances to have an "All-Star" career outcome.

The metrics I will use to build the model will be Points Per Game, Rebounds Per Game, Assists Per Game, and Player Efficiency Rating.

If I had more time I would build more models that use more metrics such as Steals Per Game and Blocks Per Game.

I will predict the chances by percentage that Shai Gilgeous-Alexander, Zion Williamson, James Wiseman, and Josh Giddey will have at least an All-Star level career outcome.

In [26]:
```python
player_data_pre_2015_draft = player_data[player_data['draftyear'] <= 2015]

cols = ["tot_reb", "points", "ast", "PER", "season", "draftyear", "Elite", "games"]
```

In [27]:
```python
season_frequency = player_data_pre_2015_draft.nbapersonid.value_counts()
players = season_frequency[season_frequency>=7].index
```

In [28]:
```python
df = player_data_pre_2015_draft[player_data_pre_2015_draft.nbapersonid.isin(players)]
```

In [29]:
```python
df = df[(df.season-df.draftyear) < 4]
```

In [30]:
```python
player_draft_dict = df.set_index("nbapersonid").draftyear.to_dict()
```

In [31]:
```python
awards["draftyear"] = awards.nbapersonid.map(player_draft_dict)
```

In [32]:
```python
target = awards.dropna(subset=["draftyear"])
```

In [33]:
```python
target = target[(target.season-target.draftyear) > 4]
```

In [34]:
```python
all_star_status = target.groupby("nbapersonid").all_star_game.sum()
```

In [35]:
```python
all_star_status = (all_star_status>0)*1.0
```

In [36]:
```python
all_star_status.value_counts()
```
Out[36]:
```
0.0    254
1.0     52
Name: all_star_game, dtype: int64
```

In [37]:
```python
ppg = (df.groupby("nbapersonid").points.sum() / df.groupby("nbapersonid").games.sum()).rename("ppg")
apg = (df.groupby("nbapersonid").ast.sum() / df.groupby("nbapersonid").games.sum()).rename("apg")
rpg = (df.groupby("nbapersonid").tot_reb.sum() / df.groupby("nbapersonid").games.sum()).rename("rpg")
perpg = df.groupby("nbapersonid").PER.mean()
```

In [38]:
```python
features = pd.concat([ppg,apg, rpg, perpg], axis = 1)
```

In [39]:
```python
data = pd.concat([features, all_star_status], axis = 1, join = "inner")
```

In [40]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
```

In [41]:
```python
X = data.drop("all_star_game", axis = 1)
y = data.all_star_game
```

In [42]:
```python
y.value_counts(normalize=True)
```
Out[42]:
```
0.0    0.830065
1.0    0.169935
Name: all_star_game, dtype: float64
```

In [43]:
```python
X_train, X_test, y_train, y_test =  train_test_split(X,y, test_size=.25, random_state=1)
X_train.head()
```
Out[43]:

nbapersonid	ppg	apg	rpg	PER
201933.0	20.407895	3.600877	10.385965	22.566667
202330.0	11.968641	3.139373	3.432056	14.800000
201961.0	6.891386	1.059925	1.992509	10.260000
200752.0	19.550000	1.858333	5.866667	16.300000
203933.0	13.678899	1.050459	4.146789	15.725000

In [44]:
```python
lr = LogisticRegression()
lr.fit(X_train, y_train)
```
Out[44]:
```
LogisticRegression()
```

In [45]:
```python
lr.score(X_train, y_train)
```
Out[45]:
```
0.8646288209606987
```

In [46]:
```python
lr.score(X_test, y_test)
```
Out[46]:
```
0.8831168831168831
```

In [47]:
```python
cv_scores = cross_val_score(LogisticRegression(), X, y, scoring = "accuracy", cv = 5)
cv_scores.mean()
```
Out[47]:
```
0.862876784769963
```

2018-2021 Prediction

In [48]:
```python
prediction_data = player_data[player_data['draftyear'] >= 2018]
```

In [49]:
```python
ppg = (prediction_data.groupby("nbapersonid").points.sum() / prediction_data.groupby("nbapersonid").games.sum()).rename("ppg")
apg = (prediction_data.groupby("nbapersonid").ast.sum() / prediction_data.groupby("nbapersonid").games.sum()).rename("apg")
rpg = (prediction_data.groupby("nbapersonid").tot_reb.sum() / prediction_data.groupby("nbapersonid").games.sum()).rename("rpg")
perpg = prediction_data.groupby("nbapersonid").PER.mean()
```

In [50]:
```python
features = pd.concat([ppg,apg, rpg, perpg], axis = 1)
```

In [51]:
```python
preds18_21 = lr.predict_proba(features)
```

In [52]:
```python
preds18_21 = pd.DataFrame(index = features.index, data = preds18_21, columns=["Not All Star", "All Star"])
```

In [53]:
```python
top10_allstar_preds = preds18_21["All Star"].nlargest(10)
top10_allstar_preds
```
Out[53]:
```
nbapersonid
1629029    0.993002
1629027    0.962881
1630163    0.899286
1629627    0.879584
1629630    0.865418
1630581    0.800510
1629028    0.768185
1630595    0.647409
1630169    0.609883
1630567    0.599530
Name: All Star, dtype: float64
```

In [54]:
```python
id2name = player_data.groupby(["nbapersonid", "player"]).size().index.to_list()
id2name = dict(id2name)
```

In [55]:
```python
preds18_21["Player Name"] = preds18_21.index.map(id2name)
```

In [56]:
```python
names = ["Shai Gilgeous-Alexander", "Zion Williamson", "James Wiseman", "Josh Giddey"]
```

In [57]:
```python
preds18_21[preds18_21["Player Name"].isin(names)]
```
Out[57]:

nbapersonid	Not All Star	All Star	Player Name
1628983	0.494967	0.505033	Shai Gilgeous-Alexander
1629627	0.120416	0.879584	Zion Williamson
1630164	0.915854	0.084146	James Wiseman
1630581	0.199490	0.800510	Josh Giddey

There is a 50.5% chance that Shai-Gilgeous Alexander will finish his career with an All-Star outcome, even though I suspect that this number would have been substantially higher if 2022 Season Data was taken into account. Without a doubt, I would advise to keep SGA at all costs.

There is an 87.95% chance that Zion Williamson will finish his career with an All-Star outcome, however, I would advise the front office that they should keep their current draft capital and not trade for this high-profile star mainly due to weight and injury issues he has had in the past. In my opinion, it would be unwise to take on that contract when these issues are present.

There is an 8.41% chance that James Wiseman will finish his career with an All-Star outcome. I would advise against signing him. Even though he is a former lottery draft pick, the metrics are clearly saying he will most likely not have the career people hoped for him coming out of Memphis.

There is an 80% chance that Josh Giddey will finish his career with an All-Star outcome. The metrics point that he will most likely have an All-Star career outcome. I would definitely advise to keep him signed for the future as him and SGA look like the cornerstones of the franchise.

Part 2 -- Predicting Team Stats

In this section, we're going to introduce a simple way to predict team offensive rebound percent in the next game and then discuss ways to improve those predictions.

Question 1

Using the `rebounding_data` dataset, we'll predict a team's next game's offensive rebounding percent to be their average offensive rebounding percent in all prior games. On a single game level, offensive rebounding percent is the number of offensive rebounds divided by their number offensive rebound "chances" (essentially the team's missed shots). On a multi-game sample, it should be the total number of offensive rebounds divided by the total number of offensive rebound chances.

Please calculate what OKC's predicted offensive rebound percent is for game 81 in the data. That is, use games 1-80 to predict game 81.

In [58]:
```python
thunder_rebs = rebounding_data[(rebounding_data['team'] == "OKC") & (rebounding_data['game_number'] <= 80)]

oreb_pct_avg = thunder_rebs['offensive_rebounds'].sum() / thunder_rebs['off_rebound_chances'].sum()

print(oreb_pct_avg)
```
```
0.288689755388714
```

ANSWER 1:

28.9%

Question 2

There are a few limitations to the method we used above. For example, if a team has a great offensive rebounder who has played in most games this season but will be out due to an injury for the next game, we might reasonably predict a lower team offensive rebound percent for the next game.

Please discuss how you would think about changing our original model to better account for missing players. You do not have to write any code or implement any changes, and you can assume you have access to any reasonable data that isn't provided in this project. Try to be clear and concise with your answer.

ANSWER 2: Take the player's offensive rebounds total for the season and subtract it from the team rebound and rebound chance total for the season then recalculate the average Offensive Rebound Percentage.

Question 3

In question 2, you saw and discussed how to deal with one weakness of the model. For this question, please write about 1-3 other potential weaknesses of the simple average model you made in question 1 and discuss how you would deal with each of them. You may either explain a weakness and discuss how you'd fix that weakness, then move onto the next issue, or you can start by explaining multiple weaknesses with the original approach and discuss one overall modeling methodology you'd use that gets around most or all of them. Again, you do not need to write any code or implement any changes, and you can assume you have access to any reasonable data that isn't provided in this project. Try to be clear and concise with your answer.

ANSWER 3: First, our offensive rebound percentage will depend on the opposing team's Defensive Rebound Percentage to more accurately predict the team's OREB Pct for the 81st game. We will need to pull data on the defensive rebound totals for the team we're facing and calculate their defensive rebound percentage. Next, we will need to take any missing players on their roster into consideration. We will adjust their defensive rebound percentage accordingly if any of their players are missing for the 81st game.