NBA_Lottery

Conor Nield

2023-10-23

NBA Draft Lottery

In the chunk below I will calculate the expected draft pick before the lottery, the chance of getting a lottery pick, and the difference between the draft lottery result and expected value.

```
import pandas as pd
df = pd.read_csv("/Users/conornield/Desktop/NBA_Draft_Lottery_8.csv")
#In this
def calculate_values(year, Rank, combinations, result, team, df):
  #Sparse each year based on the number of lottery picks and number of lottery teams
    if year >= 2019:
        picks = 4
    elif year >= 1987:
        picks = 3
    else:
      #All teams from the 1985 and 1986 draft have the same expected value and lottery chance.
      return pd.Series([3.5, 1.0, 3.5 - result])


    if year == 1989:
        teams = 9
    elif year <= 1988:
        teams = 7
    elif year <= 1995:
        teams = 11
    elif year <= 2003:
        teams = 13
    else:
        teams = 14
    #chance of lottery pick
    lc = 0
    #expected pick position
    ev = 0
    #Values below will come into play for post lottery odds
    teams_above_0 = 0
    teams_above_1 = 0
    teams_above_2 = 0
    teams_above_3 = 0
    teams_above_4 = 0
    A = 0
    B = 0
    C = 0
```

```
D = 0

#Calculating pick 1 probability
#The 1996 - 1998 draft the Toronto Raptors and Vancouver Grizzlies were ineligible for the first pi
if (year > 1995 and year < 1999):
  if (team == 'TOR' or team == 'VAN'):
    prob_first = 0
  elif (year == 1996):
    prob_first = combinations/ (1000 - 407)
  elif (year == 1997):
    prob_first = combinations / (1000 - 273)
  elif (year == 1998):
    prob_first = combinations/ (1000 - 304)
else:
  prob_first = combinations / 1000
ev += prob_first
lc += prob_first
if (year > 1995 and year < 1999):
  other_teams_after_first = df[(df["Year"] == year) & (df["Rank"] != Rank) & (df["Team"] != 'TOR') &
else:
  other_teams_after_first = df[(df["Year"] == year) & (df["Rank"] != Rank)]
other_teams_after_first_2 = df[(df["Year"] == year) & (df["Rank"] != Rank)]

# For the second pick

avg_comb_left_after_first = 1000 - sum((other_teams_after_first["Combinations"]**2) / (1000 - combi
prob_second = (combinations / avg_comb_left_after_first) * (1-prob_first)
ev += 2 * prob_second
lc += prob_second

# For the third pick
prob_third = 0
combo_removed = 0
for _, row_j in other_teams_after_first.iterrows():
  combo_j = row_j["Combinations"]
  Rank_j = row_j["Rank"]
  other_teams_after_j = other_teams_after_first_2[other_teams_after_first_2["Rank"] != Rank_j]
  p_j = combo_j / (1000 - combinations)
  for _, row_k in other_teams_after_j.iterrows():
    combo_k = row_k["Combinations"]
    p_k_given_j = combo_k / (1000 - combo_j - combinations)
    combo_removed += p_j * p_k_given_j * (combo_j + combo_k)
avg_comb_after_second = 1000 - combo_removed
prob_third += (combinations / (1000 - combo_removed)) * (1 - prob_second - prob_first)
ev += 3 * prob_third
lc += prob_third


# For the fourth pick, if applicable
prob_fourth = 0
combo_removed_third = 0
if (picks == 4):
  for _, row_j in other_teams_after_first.iterrows():
    combo_j = row_j["Combinations"]
    Rank_j = row_j["Rank"]
    other_teams_after_j = other_teams_after_first_2[other_teams_after_first_2["Rank"] != Rank_j]
    p_j = combo_j / (1000 - combinations)

    for _, row_k in other_teams_after_j.iterrows():
      combo_k = row_k["Combinations"]
      Rank_k = row_k["Rank"]
      p_k_given_j = combo_k / (1000 - combo_j - combinations)
      other_teams_after_k = other_teams_after_j[other_teams_after_j["Rank"] != Rank_k]

      for _, row_l in other_teams_after_k.iterrows():
        combo_l = row_l["Combinations"]
        p_l_given_j_k = combo_l / (1000 - combo_j - combo_k - combinations)
        combo_removed_third += p_j * p_k_given_j * p_l_given_j_k * (combo_j + combo_k + combo_l)
  avg_comb_after_third = 1000 - combo_removed_third
  prob_fourth += (combinations / (1000 - combo_removed_third)) * (1 - prob_second - prob_first - pr
  ev += 4 * prob_fourth
  lc += prob_fourth

#Post lottery odds
for _, row_m in other_teams_after_first.iterrows():
  combo_m = row_m["Combinations"]
  Rank_m = row_m["Rank"]
  p_m = combo_m/(1000 - combinations)
  if (Rank_m < Rank):
    A = 1
  else:
    A = 0
  other_teams_after_m = other_teams_after_first_2[other_teams_after_first_2["Rank"] != Rank_m]
  other_teams_after_first = df[(df["Year"] == year) & (df["Rank"] != Rank)]
  for _, row_n in other_teams_after_m.iterrows():
    combo_n = row_n["Combinations"]
    Rank_n = row_n["Rank"]
    p_n_given_m = combo_n / (1000 - combo_m - combinations)
    other_teams_after_n = other_teams_after_m[other_teams_after_m["Rank"] != Rank_n]
    if (Rank_n < Rank):
      B = 1
    else:
      B = 0

    for _, row_o in other_teams_after_n.iterrows():
      combo_o = row_o["Combinations"]
      Rank_o = row_o["Rank"]
      p_o_given_m_n = combo_o / (1000 - combo_n - combo_m - combinations)
      if (Rank_o < Rank):
        C = 1
      else:
        C = 0
      other_teams_after_o = other_teams_after_n[other_teams_after_n["Rank"] != Rank_o]
      if (picks == 4):
        for _, row_p in other_teams_after_o.iterrows():
          combo_p = row_p["Combinations"]
          Rank_p = row_p["Rank"]
          p_p_given_m_n_o = combo_p / (1000 - combo_m - combo_n - combo_o - combinations)
          if (Rank_p < Rank):
            D = 1
          else:
            D = 0
          #A + B + C + D is the number of teams with a lower rank picked in the lottery
          if ((A + B + C + D) == 0):
            teams_above_0 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
          elif ((A + B + C + D) == 1):
            teams_above_1 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
          elif ((A + B + C + D) == 2):
            teams_above_2 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
          elif ((A + B + C + D) == 3):
            teams_above_3 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
          elif ((A + B + C + D) == 4):
            teams_above_4 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
      else:
        if ((A + B + C + D) == 0):
          teams_above_0 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
        elif ((A + B + C + D) == 1):
          teams_above_1 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
        elif ((A + B + C + D) == 2):
          teams_above_2 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
        elif ((A + B + C + D) == 3):
          teams_above_3 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))

if (picks == 4):
  ev += teams_above_4 * Rank + teams_above_3 * (Rank + 1) + teams_above_2 * (Rank + 2) + teams_above
else:
  ev += teams_above_3 * Rank + teams_above_2 * (Rank + 1) + teams_above_1 * (Rank + 2) + teams_above

return pd.Series([ev, lc, result - ev])
df[["Expected_Value", "Lottery_Chance", "Change"]] = df.apply(lambda x: calculate_values(x['Year'], x['
df.to_csv("/Users/conornield/Desktop/NBA_Draft_Git.csv", index=False)
```

Total and average change by each team

```
total_change = df.groupby('Team')['Change'].sum().reset_index()
grouped_df = df.groupby('Team')['Change'].mean().reset_index()
sorted_df = grouped_df.sort_values(by='Change', ascending=False)
sorted_df.to_csv("/Users/conornield/Desktop/NBA_Draft_Lottery_Git_3.csv", index = False)
sorted_df
```

```
##            Team    Change
## 2           BOS  0.766457
## 29          WAS  0.512423
## 7           DEN  0.490651
## 6           DAL  0.455564
## 8           DET  0.399094
## 28          UTA  0.363581
## 19          NYK  0.354040
## 11          IND  0.319184
## 16          MIL  0.249934
## 25          SAC  0.165587
## 15          MIA  0.140080
## 17          MIN  0.124631
## 9           GSW  0.116588
## 23          PHX  0.106125
## 0           ATL  0.099638
## 5           CLE  0.041043
## 3           CHA -0.049397
## 27          TOR -0.052017
## 21          ORL -0.107297
## 4           CHI -0.140080
## 22          PHI -0.172801
## 12          LAC -0.195259
## 18  NOP/NOH/CHA -0.226281
## 24          POR -0.256236
## 14      MEM/VAN -0.299530
## 1           BKN -0.324350
## 20      OKC/SEA -0.324525
## 10          HOU -0.376751
## 13          LAL -0.467000
## 26          SAS -0.798057
```

```
#Import second data set of historical NBA success
df1 = pd.read_csv("/Users/conornield/Desktop/NBA_season_record.csv")
df1['Year'] = df1['Year'].astype(str).str[:4]
df1['Year'] = df1['Year'].astype(int)
df['Year'] = df['Year'].astype(int)
#Merging the two data frames
merged_df = pd.merge(df, df1, on=['Team', 'Year'])
print(merged_df.columns)
```

```
## Index(['Team', 'Year', 'Rank', 'Result', 'Combinations', 'Expected_Value',
##        'Lottery_Chance', 'Change', 'Winning Percentage', 'Playoff outcome'],
##       dtype='object')
```

```
merged_df['Year'] = merged_df['Year'].astype(int)
prev_season_df = merged_df[['Team', 'Year', 'Playoff outcome']].copy()
prev_season_df['Year'] = prev_season_df['Year'] - 1
prev_season_df.rename(columns={'Playoff outcome': 'Prev_Playoff_Outcome'}, inplace=True)
merged_df = pd.merge(merged_df, prev_season_df,  how='left', left_on=['Team', 'Year'], right_on = ['Tea
```

```
merged_df['Prev_Playoff_Outcome'] = merged_df.groupby('Team')['Playoff outcome'].shift(1)
for i in range(1, 11):
    merged_df[f'Playoff_Outcome_{i}yr'] = merged_df.groupby('Team')['Playoff outcome'].shift(-i)
correlations = {}
correlations['prev_season'] = merged_df['Change'].corr(merged_df['Prev_Playoff_Outcome'])
for i in range(1, 11):
    correlations[f'next_{i}_season'] = merged_df['Change'].corr(merged_df[f'Playoff_Outcome_{i}yr'])
for key, value in correlations.items():
    print(f"Correlation between lottery luck and {key}: {value:.4f}")
```

```
## Correlation between lottery luck and prev_season: -0.0832
## Correlation between lottery luck and next_1_season: 0.0788
## Correlation between lottery luck and next_2_season: 0.0194
## Correlation between lottery luck and next_3_season: 0.0381
## Correlation between lottery luck and next_4_season: -0.0390
## Correlation between lottery luck and next_5_season: 0.0052
## Correlation between lottery luck and next_6_season: -0.0145
## Correlation between lottery luck and next_7_season: -0.0876
## Correlation between lottery luck and next_8_season: 0.0391
## Correlation between lottery luck and next_9_season: 0.1854
## Correlation between lottery luck and next_10_season: 0.1034
```

```
merged_df['Prev_Winning_Percentage'] = merged_df.groupby('Team')['Winning Percentage'].shift(1)
for i in range(1, 6):
    merged_df[f'Winning_Percentage_{i}yr'] = merged_df.groupby('Team')['Winning Percentage'].shift(-i)
    merged_df[f'Win_Percentage_Change_{i}yr'] = merged_df[f'Winning_Percentage_{i}yr'] - merged_df['Pre
correlations = {}
for i in range(1, 6):
    correlation = merged_df['Change'].corr(merged_df[f'Win_Percentage_Change_{i}yr'])
    correlations[f'{i}_years_after'] = correlation
for years, corr_value in correlations.items():
    print(f"Correlation between lottery luck and winning percentage change {years} after draft: {corr_va
```

```
## Correlation between lottery luck and winning percentage change 1_years_after after draft: 0.1318
## Correlation between lottery luck and winning percentage change 2_years_after after draft: 0.0985
## Correlation between lottery luck and winning percentage change 3_years_after after draft: 0.0618
## Correlation between lottery luck and winning percentage change 4_years_after after draft: -0.0422
## Correlation between lottery luck and winning percentage change 5_years_after after draft: 0.0041
```