Simulating Soccer Seasons Using Bivariate Poisson Distributions

Lorenzo Lindquist and Dr. Mohammad Nooranidoost

Florida State University, Tallahassee, Florida

Abstract

- In this research project me and my mentor analyzed ways to simulate future soccer seasons in various leagues by looking at the previous season's data. Using Python 3 code we created bivariate Poisson matrices in order to find the probability of the final result of each match in a season. We then used Monte Carlo Methods to simulate these matches over and over in order to try to predict the final results of each season.

Introduction

- Soccer, more commonly known around the world as Football, seems like a simple game. Each match starts off with 11 players on each team, and both teams play for 90 minutes when it comes to league games. At the end of the match, there are three different possible outcomes for your team. You can win, draw, or lose, and this depends on how many goals you scored and how many goals your opponent scored within those 90 minutes. I became obsessed with this sport and it became a big part of my childhood and early adulthood. In recent years, as I have aged and gotten the opportunity to take upper-division classes in college, I have better understood how important statistics is and how it can be applied to almost every aspect of this world. This fall I met up with my UROP mentor Dr. Mohammad Nooranidoost, and we both found that we had a common love for the sport. While having been familiar with the game for our whole lives, we found that predicting outcomes is very difficult. We have spent these past few months finding ways to answer the question, "How can we use past data to create a simulation that accurately predicts what will happen during a soccer season?". For this research project, we chose to work with data from previous Serie A (Italian League) seasons. Keep in mind, you could use this model for most other leagues, we just chose Serie A just to keep it simple and perfect the model before moving on to leagues from other countries.

SERIE A TIM

Premier League

LaLiga

BUNDESLIGA

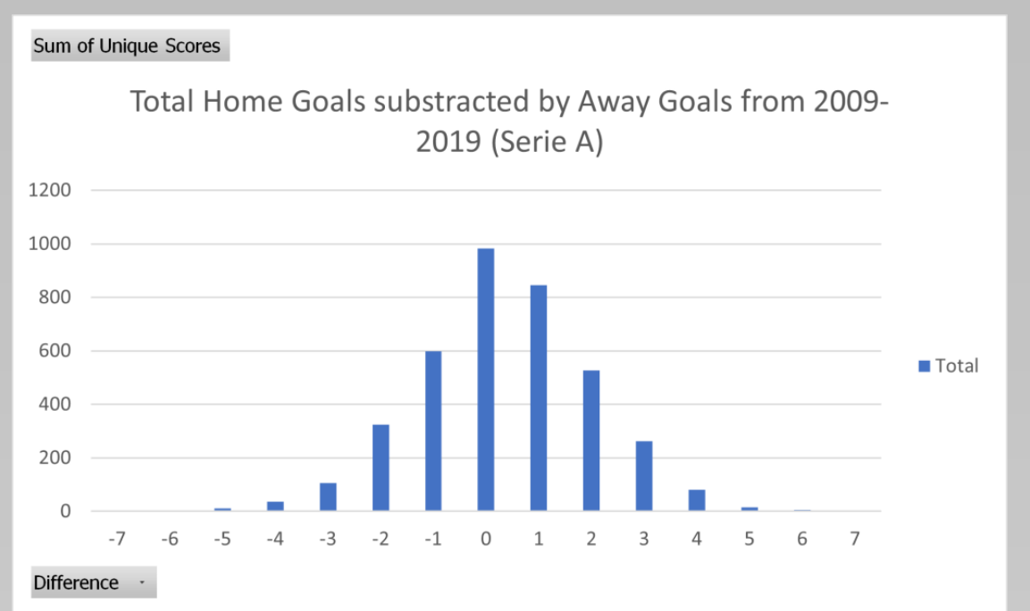
Distribution Used For the Methods

- In statistics there exists certain distributions that you can apply to specific data sets. The Poisson distribution is utilized to account for how many times an event occurs within a given amount of time. For example, you would use it in a scenario where you have to try to predict how many people are going to enter a given room within a specific period of time. As you can see, you can use this distribution to predict how many times a team scores within a 90 minute game. For example, to calculate the probability of a home team scoring just 1 goal in a game, you would need to find the mean amount of goals this team scores during a home game. You would then set lambda as being equal to the mean and x as being 1. You plug those two values into our equation below, and you should get the probability of that team scoring exactly 1 goal during a match.

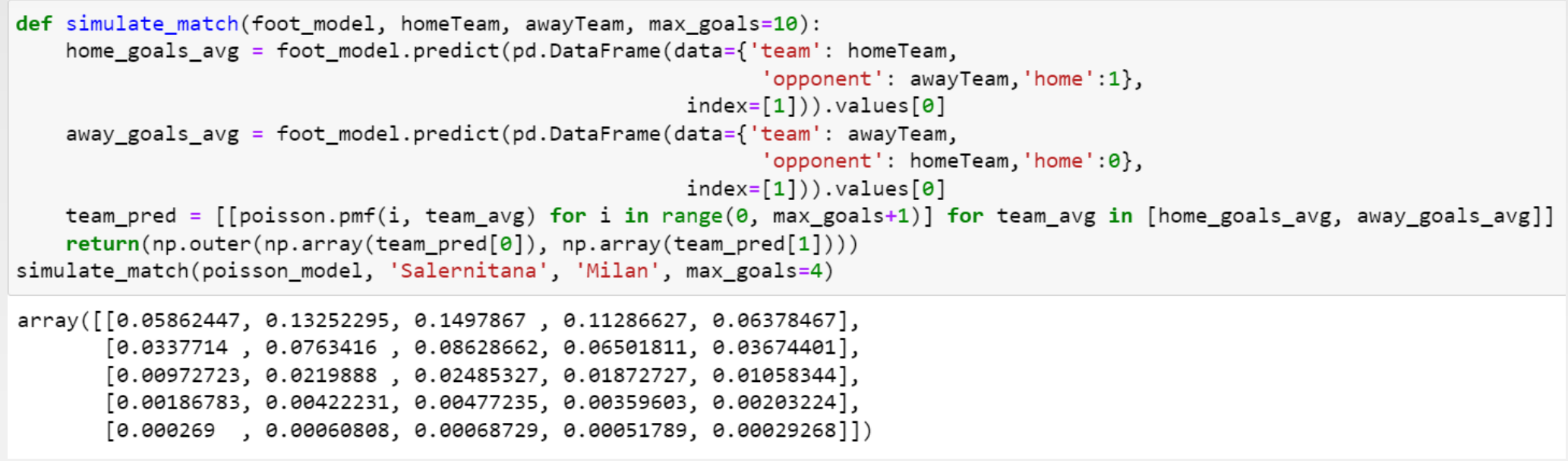
$$f(x) = \frac{\lambda^x}{x!}e^{-\lambda}$$

Methods Part I

- The primary method I used in this research project was bivariate Poisson distributions. My code analyzed previous data and found the average home goals scored and away goals scored for each team in the data set. For each matchup, you would find the probability of the home team scoring 0 up to 10 goals in a match using their home goals average and inversely, the probability of the away team finding the probability of scoring 0 up to 10 goals in a match using their away goals average. This is an important difference as looking at the graph of home goals minus away goals in each match from 2009 to 2019 we found that home teams had an advantage over away teams.



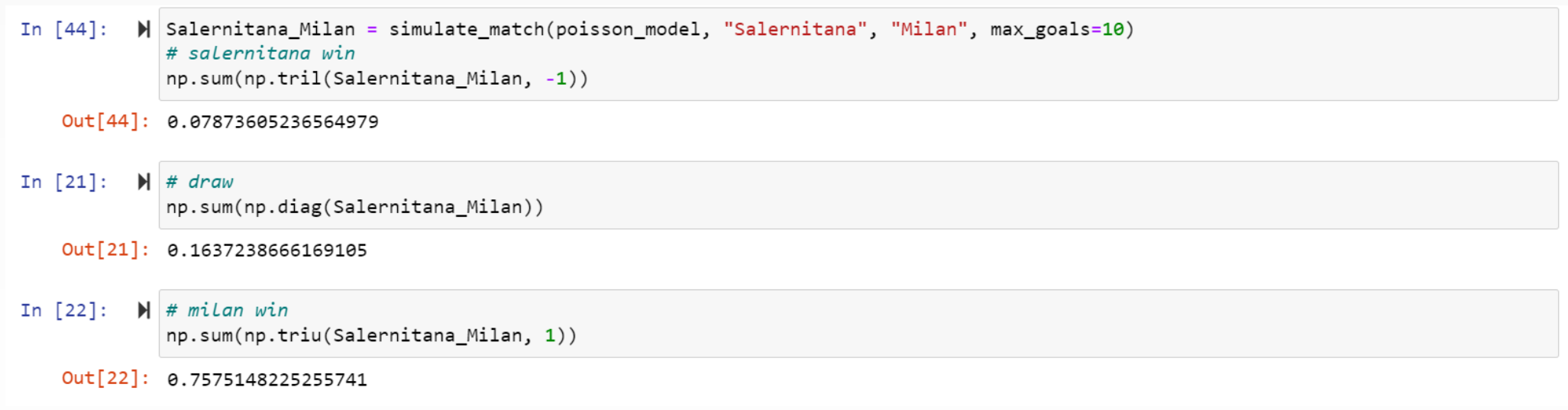
- A matrix is then created and the total probability of the home team winning the match, the two teams drawing, and the away team winning the match are calculated. Looking at this example of two teams playing a match we see our matrix. If you are unfamiliar with statistics you might be confused and not know what you're looking for. In the upper left corner would be the probability that both teams failed to score and that the game ended in a 0 to 0 draw. The box below that shows the probability that the home team Salernitana beats the away team Milan by a score of 1 to 0. The box that is diagonal to it in the upper right would inversely be the probability that the away team Milan beats the home team Salernitana by a score of 1 to 0. We only included the probability of each team scoring 0 up to 4 goals since scoring 5 to 10 goals in a match is rare. In our model we did not include the probability that a team could score 11 or more goals in a game, as there has never been a case of a team scoring 11 or more goals since it has never happened in this leagues history.



```
def simulate_match(foot_model, homeTeam, awayTeam, max_goals=10):
    home_goals_avg = foot_model.predict(pd.DataFrame(data={'team': homeTeam,
                                                           'opponent': awayTeam,'home':1},
                                                     index=[1])).values[0]
    away_goals_avg = foot_model.predict(pd.DataFrame(data={'team': awayTeam,
                                                           'opponent': homeTeam,'home':0},
                                                     index=[1])).values[0]
    team_pred = [[poisson.pmf(i, team_avg) for i in range(0, max_goals+1)] for team_avg in [home_goals_avg, away_goals_avg]]
    return(np.outer(np.array(team_pred[0]), np.array(team_pred[1])))
simulate_match(poisson_model, 'Salernitana', 'Milan', max_goals=4)
```

```
array([[0.05862447, 0.13252295, 0.1497867 , 0.11286627, 0.06378467],
       [0.0337714 , 0.0763416 , 0.08628662, 0.06501811, 0.03674401],
       [0.00972723, 0.0219888 , 0.02485327, 0.01872727, 0.01058344],
       [0.00186783, 0.00422231, 0.00477235, 0.00359603, 0.00203224],
       [0.000269  , 0.00060808, 0.00068729, 0.00051789, 0.00029268]])
```

- Before I explain the other method used, the probability of the home team winning would include the summation of all of the probabilities under the diagonal calculated, the probability of draw being the summation of all the probabilities in the diagonal, and the probability of the away team winning would be the summation of all the probabilities above the diagonal.

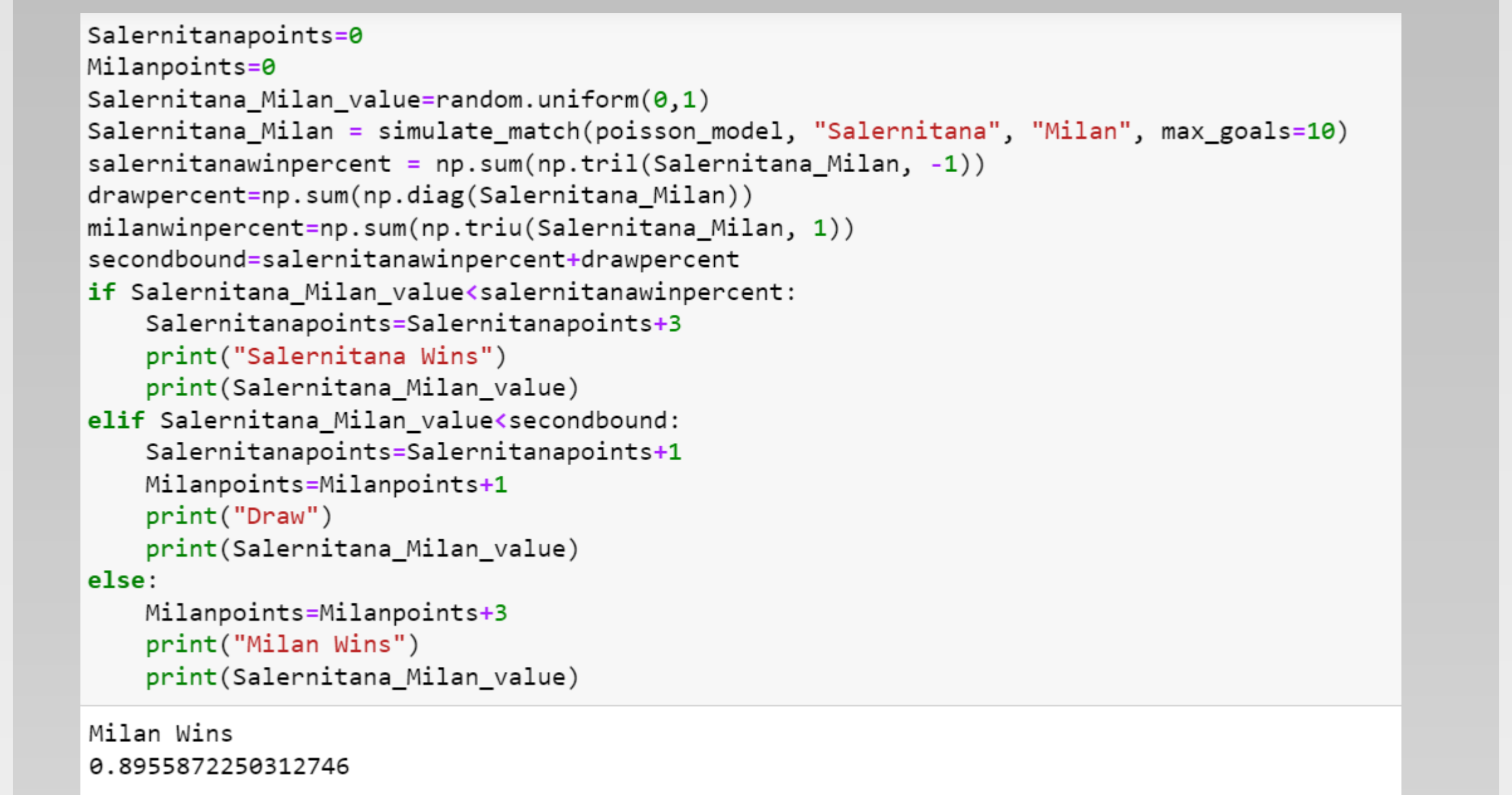


```
In [44]:  Salernitana_Milan = simulate_match(poisson_model, "Salernitana", "Milan", max_goals=10)
          # salernitana win
          np.sum(np.tril(Salernitana_Milan, -1))

Out[44]:  0.07873605236564979

In [21]:  # draw
          np.sum(np.diag(Salernitana_Milan))

Out[21]:  0.1637238666169105

In [22]:  # milan win
          np.sum(np.triu(Salernitana_Milan, 1))

Out[22]:  0.7575148225255741
```

Methods Part II

- The other method used is the Monte Carlo Method. This method uses random variables in order to predict results. In our code, our program chose a random number in a uniform distribution from a range of 0 to 1 is selected and depending on where this number falls, the match will be predicted. The correct amount of points will be awarded for each match, each team will play every other team home and away, and at the end of the season, the point totals will be calculated for the league.

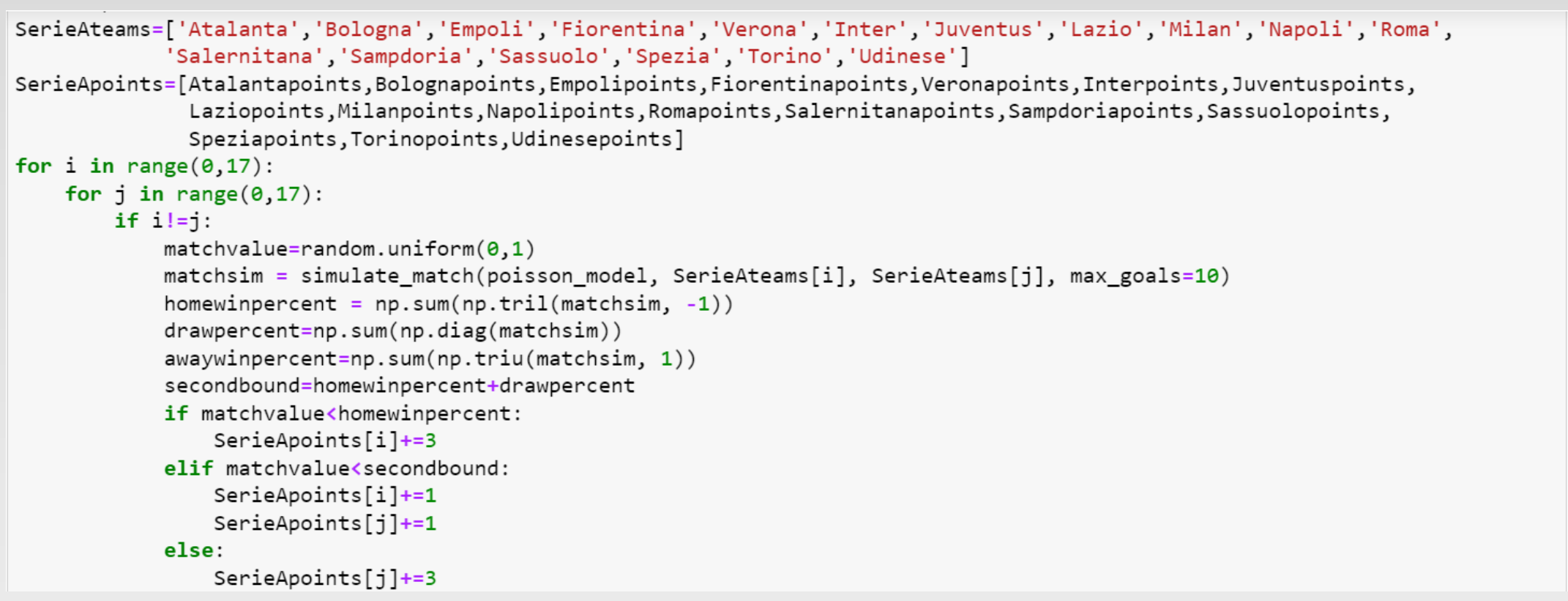
- The first piece of code under Methods Part II shows how this process is worked for just a single match.



```
Salernitanapoints=0
Milanpoints=0
Salernitana_Milan_value=random.uniform(0,1)
Salernitana_Milan = simulate_match(poisson_model, "Salernitana", "Milan", max_goals=10)
salernitanawinpercent = np.sum(np.tril(Salernitana_Milan, -1))
drawpercent=np.sum(np.diag(Salernitana_Milan))
milanwinpercent=np.sum(np.triu(Salernitana_Milan, 1))
secondbound=salernitanawinpercent+drawpercent
if Salernitana_Milan_value<salernitanawinpercent:
    Salernitanapoints=Salernitanapoints+3
    print("Salernitana Wins")
    print(Salernitana_Milan_value)
elif Salernitana_Milan_value<secondbound:
    Salernitanapoints=Salernitanapoints+1
    Milanpoints=Milanpoints+1
    print("Draw")
    print(Salernitana_Milan_value)
else:
    Milanpoints=Milanpoints+3
    print("Milan Wins")
    print(Salernitana_Milan_value)
```

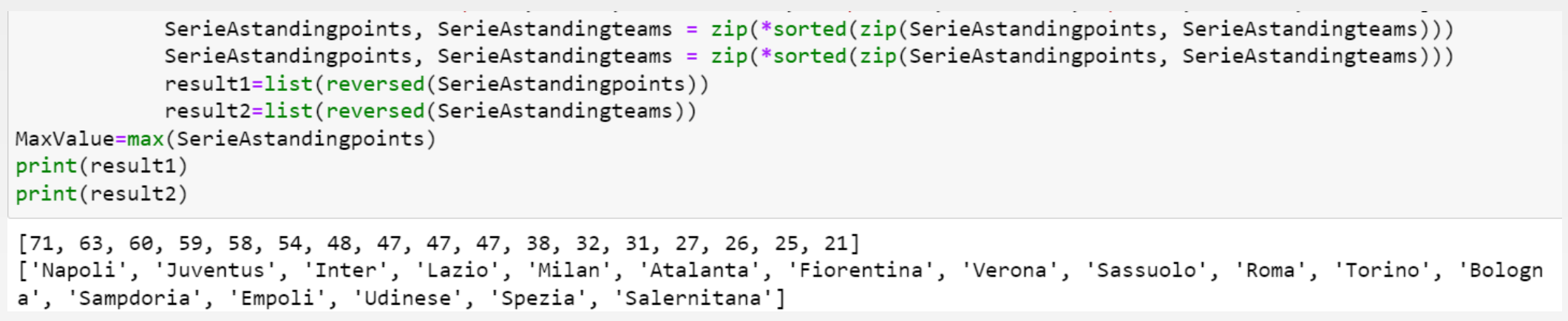
```
Milan Wins
0.8955872250312746
```

- This piece of code shows how this process was done for an entire season.



```
SerieAteams=['Atalanta','Bologna','Empoli','Fiorentina','Verona','Inter','Juventus','Lazio','Milan','Napoli','Roma',
             'Salernitana','Sampdoria','Sassuolo','Spezia','Torino','Udinese']
SerieApoints=[Atalantapoints,Bolognapoints,Empolipoints,Fiorentinapoints,Veronapoints,Interpoints,Juventuspoints,
              Laziopoints,Milanpoints,Napolipoints,Romapoints,Salernitanapoints,Sampdoriapoints,Sassuolopoints,
              Speziapoints,Torinopoints,Udinesepoints]
for i in range(0,17):
    for j in range(0,17):
        if i!=j:
            matchvalue=random.uniform(0,1)
            matchsim = simulate_match(poisson_model, SerieAteams[i], SerieAteams[j], max_goals=10)
            homewinpercent = np.sum(np.tril(matchsim, -1))
            drawpercent=np.sum(np.diag(matchsim))
            awaywinpercent=np.sum(np.triu(matchsim, 1))
            secondbound=homewinpercent+drawpercent
            if matchvalue<homewinpercent:
                SerieApoints[i]+=3
            elif matchvalue<secondbound:
                SerieApoints[i]+=1
                SerieApoints[j]+=1
            else:
                SerieApoints[j]+=3
```

- The code prints out how many points each time got and what place they ended up in..

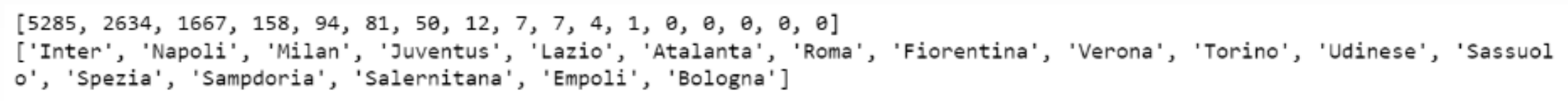


```
            SerieAstandingpoints, SerieAstandingteams = zip(*sorted(zip(SerieAstandingpoints, SerieAstandingteams)))
            SerieAstandingpoints, SerieAstandingteams = zip(*sorted(zip(SerieAstandingpoints, SerieAstandingteams)))
            result1=list(reversed(SerieAstandingpoints))
            result2=list(reversed(SerieAstandingteams))
MaxValue=max(SerieAstandingpoints)
print(result1)
print(result2)
```

```
[71, 63, 60, 59, 58, 54, 48, 47, 47, 47, 38, 32, 31, 27, 26, 25, 21]
['Napoli', 'Juventus', 'Inter', 'Lazio', 'Milan', 'Atalanta', 'Fiorentina', 'Verona', 'Sassuolo', 'Roma', 'Torino', 'Bologn
a', 'Sampdoria', 'Empoli', 'Udinese', 'Spezia', 'Salernitana']
```

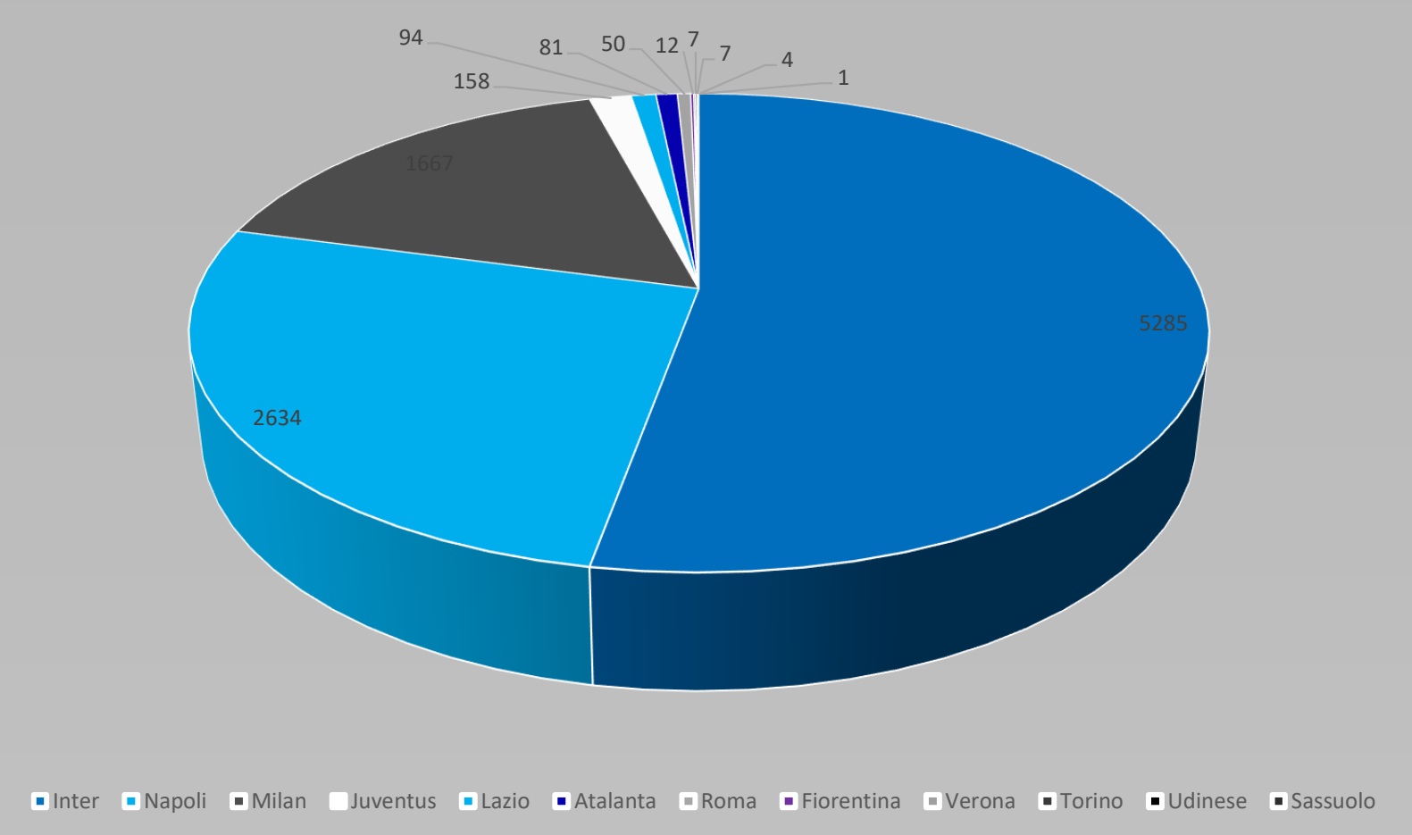
Results

- For our model, we took data from the 2021-22 Serie A Season. We ran the simulation 10000 times and we found some promising results. We found that Inter won the league 52.85% of the time, Napoli won the league 26.34% of the time, Milan won 16.67% of the time. That means that 95.86% of the time, one of those three teams won the league in our model. Looking at the final results of that season, those were the three teams that did the best, with Milan being first, Inter just barely missing out on second, and Napoli being third.



```
[5285, 2634, 1667, 158, 94, 81, 50, 12, 7, 7, 4, 1, 0, 0, 0, 0, 0]
['Inter', 'Napoli', 'Milan', 'Juventus', 'Lazio', 'Atalanta', 'Roma', 'Fiorentina', 'Verona', 'Torino', 'Udinese', 'Sassuol
o', 'Spezia', 'Sampdoria', 'Salernitana', 'Empoli', 'Bologna']
```

Every League Winner for our 10000 simulated seasons



Conclusions and Future Research

- While me and my research mentor made progress during the first few months of this research project, we still have to figure some aspects of our model out. If you paid attention to the code you will see that we only had 17 teams in our table instead of 20. One of the next steps is to figure out how to account for newly promoted teams. These newly promoted teams do not have any data from the previous season so trying to predict how they are going to perform is going to be a difficult task.

- Another aspect of this research project I am going to try to work on is to see if any other statistics other than goals are useful for predicting matches. I am going to run variable regressions with each teams shots, shots on goal, possession, and expected goals to see if they are significant in predicting the outcome of a match.

- For this project we only used the data from the previous season in order to make predictions for the following season. This can lead to same inaccurate predictions as some teams can improve or regress depending on what happens between and during seasons. Me and my research mentor are going to attempt to only include data from the previous 10 matches in order to make sure that the data we use is newer and more relevant for our model.

- The model that we are creating can only be used for a league season. Once me and my mentor feel like we have a proper adequate model for leagues, we are going to attempt to create models for knockout competitions such as the World Cup, Champions League, and FA Cup.

References

Sheehan, David. "Predicting Football Results with Statistical Modelling." *dashee87.Github.io*, 4 June 2017, https://dashee87.github.io/football/python/predicting-football-results-with-statistical-modelling/.

Whitaker, Gavin. *The Bivariate Poisson Distribution and Its Applications ... - Github Pages*. 5 May 2011, https://gawhitaker.github.io/project.pdf.

Acknowledgments

•Thank you to the UROP program for providing us with a UROP materials grant to work on this project.