Aryan Kumawat (Pd 4) and Trishal Muthan (Pd 2)

Fast.AI 2: NBA Game Simulator

For Fast.ai pt. 2, we created an NBA Game Simulator using a public dataset found on Kaggle. The model we created can be used to predict who wins a game between any two NBA teams using historical data of NBA games from 2014-2018.

Our code and what it does:

We imported 4 different libraries for us to use:

```
import random as rnd
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

We then read in the data from the dataset using Pandas and separated the data into two dataframes, one for the Boston Celtics and the other for the Cleveland Cavaliers, the two teams we would be using in the simulation.

```
In [4]: import random as rnd
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd

In [6]: gamedf = pd.read_csv('nba.games.stats.csv')

In [9]: bosdf = gamedf[gamedf.Team == 'BOS']
        cldf = gamedf[gamedf.Team == 'CLE']
```

We then gathered some of the key statistics that we would use later on:

```
bosmeanpts = bosdf.TeamPoints.mean()
clmeanpts = cldf.TeamPoints.mean()
bosdpts = bosdf.TeamPoints.std()
clsdpts = cldf.TeamPoints.std()

bosmeanopp = bosdf.OpponentPoints.mean()
clmeanopp = cldf.OpponentPoints.mean()
bossdopp = bosdf.OpponentPoints.std()
clsdopp = cldf.OpponentPoints.std()
```

These were our simulation methods. gameSimulation() handled the actual simulation of a game using gaussian distributions on the data and returned either 1, -1, or 0 corresponding to win, lose, or indecisive (tie) respectively. gamesSimulator() simulated the given number of games and returned a percentage value for the amount won, lost, and tied.

In [32]:
```
def gameSimulation():
    BOSScore = (rnd.gauss(bosmeanpts,bosdpts)+ rnd.gauss(clmeanopp,clsdopp))/2
    CLScore = (rnd.gauss(clmeanpts,clsdpts)+ rnd.gauss(bosmeanopp,bossdopp))/2
    if int(round(BOSScore)) < int(round(CLScore)):
        return -1
    if int(round(BOSScore)) > int(round(CLScore)):
        return 1
    else:
        return 0
```

In [35]:
```
def gamesSimulator(numOfSimulations):
    teamBoswin = 0
    teamClewin = 0
    indecisive = 0
    for i in range(numOfSimulations):
        game = gameSimulation()
        if game == 1:
            teamBoswin +=1
        if game == -1:
            teamClewin +=1
        else:
            indecisive +=1
    percentage_bos = "{:.0%}".format(teamBoswin/(teamBoswin+teamClewin+indecisive))
    print('BOS Win ', percentage_bos)
    percentage_cle = "{:.0%}".format(teamClewin/(teamBoswin+teamClewin+indecisive))
    print('CLE Win ', percentage_cle)
    indecisive_percent = "{:.0%}".format(indecisive/(teamBoswin+teamClewin+indecisive))
    print('Indecisive ', indecisive_percent)
```

A mock output of the simulation run on a Jupyter notebook is shown below:

[30]:
```
gamesSim(10000)

BOS Win  54%
CLE Win  43%
Indecisive  3%
```

This shows that in 10000 games between the Boston Celtics and the Cleveland Cavaliers, the Cavaliers would win 43%, the Celtics would win 54%, and the result is indecisive (or a tie) for 3% of them.

How it went:

We had a lot of fun working on this project because both of us are really big NBA fans and felt that applying some of our knowledge to this area would be something we'd get a lot out of. Both of us were relatively familiar with Pandas so there wasn't too much of an issue from that standpoint. The gameSimulation function was a bit annoying, as it was difficult to get something that actually modeled NBA games well, rather than outputting random values. It took a lot of double checking with the data and experimenting, but with help from online, we were able to come to some formulas that we were comfortable with and felt were able to accurately model NBA games between particular teams. The gamesSimulator method wasn't too difficult as all we had to do was run through the necessary simulations and output result values. The result was pretty cool. We were able to see how many times a team would win if they played each other a 1000 times, and we thought it could be easily applicable to the real NBA and it's games; that is probably what sports bettings and the NBA apps do in a similar fashion. One thing we think we both gained throughout the entire experience was helpful exposure to Jupyter. Jupyter notebooks are not something either of us are all that familiar with, so learning all the different

features and capabilities of Jupyter notebooks is something we will both take from this assignment.

The Kaggle dataset can be found here:
https://www.kaggle.com/ionaskel/nba-games-stats-from-2014-to-2018?select=nba.games.stats.csv
The dataset had every game played between 2014-2018 of each team, and had a variety of both categorical and quantitative data regarding the game (i.e Opponent, WinorLoss, OpponentPoints, FieldGoals, FreeThrows, Turnovers, etc).