Econ 524 Colley Matrix Assignment

For this assignment, you may choose to work alone or work in a group of 2 people. I encourage working in pairs because debugging is often easier with 2 pairs of eyes looking at the code. If you work with a partner, please submit only one project with both names included.

For this assignment, we will work with data on historical outcomes of college football games and implement the Colley ranking method to rank teams in a given year. Because college football teams play only a small fraction of the total teams, a ranking is necessary to determine who plays in the championship game or who advances to the playoffs. A simple ranking by winning percentage would be misleading because there is wide variation in the strength of schedule. That is, some teams play a much more difficult schedule than others because of the conference they play in. Furthermore, since teams have some leeway to choose their out-of-conference games, teams would have incentives to schedule easy games for the out-of-conference schedule if the ranking does not control for strength of schedule. With that in mind, Colley developed a ranking method that does account for strength-of-schedule when ranking teams. The PDF file on Canvas provides the full details of the Colley method, although it is not necessary to read it if you follow the directions provided here.

For this project, you will implement the Colley matrix method to rank teams for the 2010 season.

The Data

The data is available in the file ncaa.pkl. You can load this data frame using the command:

```
pd.read_pickle("ncaa.pkl")
```

The data frame has 4 columns. The first is of type str and has the 120 team names. The second is int type and has the number of wins for that team. The third is int type and has the number of losses for that team. The fourth column is array type where each row contains an array of indices of the other teams played by that team. For example, if the row 0 of this column was the array [4 8 2], it would mean the team in row 0 played 3 games, one against the team in row 4, one against the team in row 8, and one against the team in row 2. Sometimes teams play each other more than once, in which case the same index may appear more than once in the opponents array. For example, if row 0 of this column was the array [4 8 2 8], it would mean the team in row 0 played 4 games, one against team 4, one against team 2, and two against team 8.

The Colley Matrix Method

You need to construct the Colley matrix (see section 6.1 of the PDF file). The Colley matrix is a square matrix with dimension equal to the number of teams. A 2-dimensional NumPy array can be used to represent this matrix. The diagonal entries (i,i) in the matrix are (2+num games played by team i). The off diagonal entries (i,j) are set to (-1*(# games played by team i against team j)). So, each row should sum to 2.

You also need to construct a "b" vector (represented as a 1-dimensional NumPy array) where the i-th element is:

1 + (team i num wins – team i num losses)/2

Then, solve the linear system Ax=b where A is the matrix and b is the vector. The solution is the score for each team where a higher score is better.

Section 6.3 shows an example with 2 teams and one game where the first team beat the second team in that game. There is also a 5 team example shown at the end of Section 6.3.

Python Implementation Details

In Python, we can represent matrices and vectors with the NumPy array data structure. The vector can be represented as a 1-dimensional array and the matrix with a 2-dimensional array. The same functions for 1-dimensional arrays also work with 2-dimensional arrays with slight differences in syntax. For example, here is how you could create an array (or matrix) of all zero's with 5 rows and 6 columns (note double parentheses):

```
A=np.zeros((5,6))
```

And here is how you could get the value of the element in the row=2, column=3 (using standard python indexing that starts at 0):

```
A[2,3]
```

In order to treat a 2-dimensional array as a matrix, there are some other useful functions that are part of NumPy and that you will need to use for this project:

np.linalg.inv(A) #computes inverse of A (where A is a 2-dimensional array with same number of rows and columns)

np.matmul(A,B) #A is an n x k array and B is a k x m array, computes A*B (matrix multiplication)

What to Turn In

You must follow the directions here EXACTLY. Please read carefully.

Your solution should be in a single file called colley.py. You can use pandas and numpy, but no other modules.

In that file, you should create a function called colley_rank which takes one argument, a data frame, that has the format described above in "The Data". The function should compute the Colley rankings and return a list with 3 elements.

The first element of the return list should be a data frame which has two columns named "team" and "score". The first column should be the team name and the second column should be the Colley score. This data frame should be sorted so that the best team (highest score) is in row 0 and the worst team (lowest score) in the last row. The sort_values function for data frames will help with this.

The second element of the return list should be the Colley matrix described above.

The third element of the return list should be the b-vector used in the Colley method and described above.

You can test your function using the data frame in ncaa.pkl but you do NOT need to turn in the pkl file and your function should NOT read that file, it should use the data frame passed in as an argument. Your function should work on any data passed in the correct format, not just the 2010 data.

Additional Tips

Hint: Here is how I approached the problem (written in “pseudocode”). There are other ways to do it. Each line in the pseudocode should translate to one line of Python code.

```
Initialize matrix A to be all 0’s (number of rows and columns equal number of teams)
Initialize b-vector to be all 0’s (number of elements equals number of teams)
Loop for i= 1 to #teams:
    Set diagonal entry for team i  A[i,i] based on formula
    Set b[i] based on formula
    Loop for j in array of opponents of team i:
        Subtract 1 from A[i,j]  #(see below for further explanation of this)
    End j loop
End i loop
Solve linear system
Combine team name and Colley score into a new data frame
Sort that data frame
Create return list
```

# The idea here is that A has already been initialized to 0, so when we find two teams have played and we subtract 1, we are setting C[i,j] to be 0 – 1 = - 1 which is what we want.