Senior Thesis in Mathematics

An Analysis of Division 3 Men's College Soccer Rankings: A Data-Driven Look at Systemic Bias

Nicholas Mikhail

Advisor: Prof. Gabe Chandler

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

April 2025

Student Nicholas Mikhail

1 Quantifying the Impact of Imbalanced Groups in FIFA Women's World Cup Tournaments (1991–2019)

1.1 Motivation

Over the history of the FIFA Women's World Cup, there has been a persistent pattern of extremely unbalanced group stage matches, games where one team dominates another with scoring margins in the double digits. These lopsided outcomes often trace back to how the tournament groups were seeded and assigned. In particular, the grouping procedures frequently resulted in substantial differences in competitive strength between groups, creating unfair advantages or disadvantages for certain teams.

Lapre and Palazollo [7] sought to rigorously evaluate this issue by introducing a mathematical framework to quantify the strength of teams and groups. Their primary objective was to assess how group imbalance affects fairness and advancement outcomes in tournaments spanning from 1991 to 2019.

1.2 Team Rating Methodology

To assess team strength, the authors employed a least squares regression approach that assigns a numerical rating to each team in a given tournament. These ratings are based on observed goal differentials between teams. The central idea is to estimate how much stronger or weaker a team is relative to others, using actual match results as evidence.

The optimization problem is framed as follows:

$$\min \sum_{(i,j)\in S(t)} \left((r_i^t - r_j^t) - m_{ij}^t\right)^2 \quad \text{s.t.} \quad \frac{1}{N_t}\sum_{i=1}^{N_t} r_i^t = 0 \tag{1}$$

Here:

- r_i^t is the estimated rating for team i in tournament t.
- m_{ij}^t is the observed goal differential between teams i and j in tournament t.
- N_t is the total number of teams in tournament t.

The model minimizes the squared error between predicted scoring margins (based on the rating differences) and the actual outcomes. The constraint ensures that the average team rating in each tournament is zero, thus normalizing all ratings and avoiding inflation or deflation.

The authors used Excel Solver to estimate these team ratings for each tournament.

1.3 Measuring Group Strength and Competitiveness

To determine how strong or weak each group was, the authors introduced two key metrics:

Group Strength:

$$g_s(G_t) = \frac{1}{4}\sum_{i\in G} r_i^t$$

This represents the average rating of the four teams in a group. Groups with higher values were composed of stronger teams overall.

Opponent Strength:

$$g_{\text{opp}}(r_i^t) = \frac{1}{3}\sum_{j \in G_i} r_j^t$$

This measures the average strength of the three group stage opponents for team i.

These calculations provided both a macro level view of group difficulty and a micro level assessment of each team's relative group challenge.

Their findings revealed that every Women's World Cup from 1991–2019 had an average goal margin above 2 goals per game, a benchmark used to define whether a game was competitive. This supports the notion that group assignments often resulted in imbalanced competition.

1.4 Logistic Regression: Predicting Quarterfinal Success

To investigate how various factors influenced the probability of a team reaching the quarterfinals, the authors developed a logistic regression model:

$$\ln\left(\frac{\Pr(QF_{it}=1)}{1-\Pr(QF_{it}=1)}\right) = \beta_0 + \beta_1 N_t + \beta_2 HC_{it} + \beta_3 r_{it} + \beta_4 g_{\text{opp}}(r_{it}) + e_{it} \tag{2}$$

Where:

- $QF_{it} = 1$ if team i reached the quarterfinals in year t; 0 otherwise.
- N_t = number of teams in the tournament.
- $HC_{it} = 1$ if the tournament was played on the team's home continent; 0 otherwise.
- r_{it} = team i's rating.
- $g_{\text{opp}}(r_{it})$ = average strength of team i's group opponents.

Interpretation of Coefficients:

- $\beta_1 < 0$: Teams in larger tournaments face stiffer competition and are less likely to advance.
- $\beta_2 > 0$: Home continent teams are more likely to reach the quarterfinals, suggesting geographical advantages.
- $\beta_3 > 0$: Stronger teams (higher ratings) are more likely to advance.
- $\beta_4 < 0$: Facing stronger opponents in the group stage reduces a team's likelihood of advancing.

The pseudo-R^2 values from the models were high, indicating that the model explained a large proportion of the variance in quarterfinal advancement.

1.5 Logistic Function and Visualization

Using the fitted model, the authors produced predictive visualizations:

- **Scenario 1:** An average team's chance of reaching the quarterfinal depending on opponent group strength (Max, Equal, Min).
- **Scenario 2:** Probability curves based on tournament size (12, 16, 24 teams) and continent (home vs away).

These charts showed that:

- Stronger group opponents reduce the probability of advancement.
- Teams playing on their home continent have a higher chance of advancing than those playing abroad.
- Smaller tournaments benefit average teams due to fewer competitive barriers.

1.6 On Table 3

While the paper presents Table 3 as a distinct contribution, it appears to mainly omit earlier tournaments from the analysis, which may help isolate the effect of structural expansion (from 12 to 24 teams). However, the model structure and logic remained largely consistent.

1.7 Replication in R

To validate the authors' methodology, I replicated their least squares team rating model in R for the 2014, 2018, and 2022 Men's World Cups.

- Each match was encoded as a row in a model matrix with +1 and -1 indicators for each team.
- The goal differential served as the response variable.
- Linear regression was performed, with the constraint built into the model to center ratings around zero.

Results:

- Intercepts were all close to zero, consistent with the constraint.
- The model performed well for teams that played multiple games.
- Teams eliminated in the group stage with only 3 matches often had skewed ratings due to small sample sizes or outlier results.
- The 2022 World Cup showed slightly less alignment between coefficients and team outcomes.
- The average team rating in 2022 was slightly positive, possibly due to Qatar's favorable match setup as host.

1.8 Conclusion

This paper, along with the replication, demonstrates how mathematical modeling can illuminate structural inequities in tournament design. Least squares ratings provide a strong baseline for assessing group strength, and when paired with logistic regression, offer insight into how structural factors impact competitive fairness. With minimal data, the methodology accurately estimated team quality and revealed how group imbalance can affect advancement probabilities, especially in growing tournaments.

2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a pivotal optimization method used primarily in machine learning and deep learning for training a wide range of models, including linear regression, logistic regression, and neural networks [5]. Bottou explains how at its core, SGD is an iterative method for optimizing an objective function, typically aimed at finding the minimum of a function. The function represents the cost or loss of the model, which the algorithm attempts to minimize. [1]

To understand SGD, one must first grasp the concept of Gradient Descent. Gradient Descent is an optimization algorithm used to minimize functions by iteratively moving towards the steepest descent as defined by the negative of the gradient. In mathematical terms, if f is a differentiable function, the formula for updating the parameters in Gradient Descent is given by:

$$\theta = \theta - \eta \cdot \nabla_\theta f(\theta) \tag{3}$$

where θ represents the parameters of the model, η is the learning rate, and $\nabla_\theta f(\theta)$ is the gradient of the function f with respect to θ.

While Gradient Descent updates parameters using all training data to compute the true gradient, Stochastic Gradient Descent updates parameters using only a single data point (or a subset of data points) at each iteration [1]. This randomness significantly reduces the computational burden, making SGD particularly useful for large datasets.

The update rule for SGD can be expressed as:

$$\theta = \theta - \eta \cdot \nabla_\theta f_i(\theta) \tag{4}$$

where f_i is the cost associated with a single randomly chosen sample or a small batch of samples from the training dataset.

In SGD, the gradient is a vector of partial derivatives; it points in the direction of the steepest increase of the function. By moving in the opposite direction, we aim to find the minimum of the loss function more efficiently.

Given a function $f(\theta)$, where θ represents the parameters of a model, the gradient of f at θ is given by:

$$\nabla f(\theta) = \left[\frac{\partial f}{\partial \theta_1}, \frac{\partial f}{\partial \theta_2}, \dots, \frac{\partial f}{\partial \theta_n}\right]^T \tag{5}$$

In SGD, the update rule for the parameter vector θ at iteration t is:

$$\theta_{t+1} = \theta_t - \eta_t \nabla f_{i_t}(\theta_t) \tag{6}$$

where η_t is the learning rate at time t, and $\nabla f_{i_t}(\theta_t)$ is the gradient estimated using a randomly selected subset of data.

2.1 Impact of the Gradient

The magnitude and direction of the gradient have significant implications:

- **Magnitude**: The size (or magnitude) of the gradient vector gives an indication of how steep the slope is at a particular point. A large magnitude implies a steep slope, suggesting that a significant change in the function value can be achieved by making a relatively small change in the parameters. Conversely, a small magnitude indicates a flatter slope. Large gradients can cause the updates to overshoot the minimum, while very small gradients can slow down the learning process.

- **Direction**: The gradient points in the direction of steepest increase of the function being optimized. To find the minimum of this function, one would move in the opposite direction of the gradient. This approach underlies gradient descent techniques, where parameters are adjusted in the opposite direction of the gradient of the loss function at each step. Incorrect gradient calculations can lead to diverging from the optimal path. [5]

2.1.1 Proof that the Gradient Points in the Direction of Steepest Ascent

Theorem

Let $f(\mathbf{x})$ be a differentiable scalar function. The gradient of f, denoted $\nabla f(\mathbf{x})$, points in the direction of the steepest ascent of the function.

Proof

Let $f(\mathbf{x})$ be a differentiable function, where $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in R^n$, and the gradient of f is given by:

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

The gradient provides directional information about how the function changes at $\mathbf{x}$.

Directional Derivative

The *directional derivative* of f at $\mathbf{x}$ in the direction of a unit vector $\mathbf{u}$ is defined as:

$$D_{\mathrm{u}} f(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{u}) - f(\mathbf{x})}{h}$$

This represents the rate of change of f at $\mathbf{x}$ in the direction of $\mathbf{u}$. It can be computed as:

$$D_{\mathrm{u}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{u}$$

Where $\nabla f(\mathbf{x}) \cdot \mathbf{u}$ is the dot product of the gradient and the direction vector $\mathbf{u}$.

Maximizing the Directional Derivative

To find the direction of steepest ascent, we need to maximize the directional derivative $D_{\mathrm{u}} f(\mathbf{x})$ subject to the constraint that $\mathbf{u}$ is a unit vector, i.e., $||\mathbf{u}|| = 1$.

The directional derivative is:

$$D_{\mathrm{u}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{u} = ||\nabla f(\mathbf{x})|| ||\mathbf{u}|| \cos \theta$$

where θ is the angle between $\nabla f(\mathbf{x})$ and $\mathbf{u}$, and $||\nabla f(\mathbf{x})||$ is the magnitude of the gradient.

The directional derivative is maximized when $\cos \theta = 1$, i.e., when $\mathbf{u}$ is in the same direction as the gradient. Therefore, the maximum value of $D_{\mathrm{u}} f(\mathbf{x})$ occurs when $\mathbf{u} = \frac{\nabla f(\mathrm{x})}{||\nabla f(\mathrm{x})||}$, and the maximum rate of ascent is $||\nabla f(\mathbf{x})||$.

Conclusion

The gradient $\nabla f(\mathbf{x})$ points in the direction of the steepest ascent, and its magnitude represents the maximum rate of increase in the function. The gradient vector $\nabla f(\mathbf{x})$ always points in the direction in which the function f increases the fastest. This is because the directional derivative is maximized when the direction vector u is aligned with the gradient, and its magnitude represents the rate of increase in the function along that direction

2.2 Learning Rate Considerations

The learning rate η is a critical hyper parameter in SGD. It determines the size of the steps taken towards the minimum.

Choosing the Initial Learning Rate

Selecting an appropriate initial learning rate is crucial for the convergence of SGD. Too large a learning rate can cause the model parameters to oscillate around the minimum or diverge, whereas too small a learning rate makes the convergence slow, impacting training efficiency and effectiveness. Ketkar dives into how vital selecting a good learning rate is and the different implications it can have. [5]

Adaptive Learning Rates

To address the challenges associated with a constant learning rate, various adaptive learning rate techniques have been developed:

- **Learning Rate Decay**: Gradually reduces the learning rate over time, which helps to stabilize the training as it approaches a minimum.

$$\eta_t = \frac{\eta_0}{1 + \delta t} \tag{7}$$

  where δ is the decay rate, and t is the iteration number.

- **Momentum based SGD**: Incorporates the concept of momentum by using an exponential moving average of the gradients to update the parameters. [5]

$$v_t = \gamma v_{t-1} + \eta_t \nabla f_{i_t}(\theta_t) \tag{8}$$

$$\theta_{t+1} = \theta_t - v_t \tag{9}$$

  where γ is the momentum factor.

2.2.1 Decaying Learning Rate

Decaying learning rate is used for faster convergence at the start. In the initial stages of training, a larger learning rate can be beneficial because it allows the model to quickly converge towards the global minimum or a good local minimum. Early on, when the parameters are far from optimal, large steps (high learning rate) help in making rapid improvements.

Stability Near Minimum As the model parameters begin to approach the optimal values, continuing with a large learning rate can lead to overshooting the minimum or oscillating around it. Reducing the learning rate helps in taking smaller, more precise steps, which increases the likelihood of settling into the minimum rather than overshooting it.

2.3 Convergence of Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is widely known for its effectiveness in optimizing non convex functions and large datasets due to its ability to make frequent updates based on small batches of data. However, the convergence of SGD, particularly its theoretical guarantees, depends on several conditions.

The paper by Shapiro and Wardi [12] provides a formal analysis of SGD convergence, establishing necessary conditions under which SGD converges to a stationary point of the objective function. The authors show that if the learning rate η_t diminishes over time and the gradient estimates become more accurate, then SGD converges to a stationary point with probability one. The general form of the update rule for SGD is given by:

$$\theta_{t+1} = \theta_t - \eta_t \nabla f_{i_t}(\theta_t)$$

where η_t is the learning rate at time t and $\nabla f_{i_t}(\theta_t)$ is the gradient of the loss function based on a randomly chosen sample i_t. As $t \to \infty$, the learning rate must satisfy certain properties, such as:

$$\sum_{t=1}^{\infty} \eta_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty$$

These conditions ensure that while the learning rate decreases over time, it does not do so too quickly, which would prevent convergence. The learning rate must be large enough in the early stages to make significant progress, but small enough in the later stages to ensure fine tuned convergence near the optimal solution.

2.3.1 Generalization of Gradients

One challenge in applying gradient based methods, particularly for non differentiable functions, is the need to generalize the concept of gradients. For non differentiable convex functions, SGD uses subgradients, which generalize the idea of gradients to functions that are not smooth. [12]

A subgradient of a convex function f at point θ is any vector g such that:

$$f(\theta') \geq f(\theta) + g^T(\theta' - \theta) \quad \text{for all} \quad \theta' \in R^n$$

Subgradients allow SGD to be applied to a broader class of functions, enabling the algorithm to make updates even when the function is not differentiable at certain points. This is particularly useful when training models like ReLU based neural networks, where the activation function introduces non smoothness in the objective function.

2.3.2 Proof of Convergence

The convergence of SGD can be formally proven under the assumption that the objective function is convex and continuously differentiable, or subdifferentiable in the case of non smooth convex functions. Under these assumptions, and using the learning rate schedule defined above, the expected value of the gradient norm diminishes over time:

$$\lim_{t\to\infty} E[||\nabla f(\theta_t)||^2] = 0$$

This implies that SGD converges to a stationary point where the gradient is zero, i.e., $\nabla f(\theta) = 0$, which is either a local or global minimum of the function, depending on the properties of the objective function. In the context of training models such as autoencoders or neural networks, this result ensures that SGD will eventually find a set of parameters that minimizes the loss function. [12]

2.3.3 Practical Implications for Autoencoder Training

In the context of training autoencoders, the convergence analysis of SGD is critical for ensuring that the model parameters converge to a solution that minimizes reconstruction error. The choice of learning rate and the application of momentum or learning rate decay are key factors in achieving stable and efficient convergence. Moreover, the use of subgradients enables SGD to handle the non smooth nature of activation functions like ReLU, which are commonly used in autoencoders.

Overall, the theoretical results from Shapiro and Ward [12] provide a foundation for understanding why SGD is an effective optimization technique for training neural networks, including autoencoders, even when the optimization landscape is complex and non convex.

3 Stochastic Gradient Descent (SGD) for Team Ratings in the World Cup

The goal of this implementation was to assign coefficient ratings to teams participating in the World Cup, based on their match performances, using the Stochastic Gradient Descent (SGD) algorithm. This method evaluates team performances based on the goal differences between home and away teams, iteratively updating their ratings over multiple matches[8].

3.1 Initializing Team Ratings

The algorithm begins by extracting unique team names from the match dataset (both home and away teams) and assigns an initial rating of 0 to each team. This is achieved by creating a vector, `team_ratings`, where each team starts with the same rating. Initially, we assume that all teams are of equal strength, and the ratings will adjust as the model processes the match results [8]. The goal of this was to able to see the vailidty of SGD and how it assigns ratings to teams, and to investigate if it can be a suitable and potential option for future use cases based on how accurately it gave teams a rating compared to their actual performances.

3.1.1 Iterative Updates Using SGD

The algorithm then performs multiple iterations over the match data, which is shuffled at the start of each iteration to ensure stochasticity. The shuffling simulates the random nature of the stochastic gradient descent, providing a variety of data orders for better generalization.

In each iteration, for every match:

- The **home team** and **away team** are identified.
- The **actual goal difference** (home team score minus away team score) is retrieved, which serves as the target variable.
- The **predicted goal difference** is calculated using the current ratings of the home and away teams.
- The **error** is computed as the difference between the actual and predicted goal difference.

3.2 SGD Update Rule

The team ratings are updated using the following rule:

$$\text{new\_rating}_{\text{home\_team}} = \text{old\_rating}_{\text{home\_team}} + \eta \times \text{error}$$
$$\text{new\_rating}_{\text{away\_team}} = \text{old\_rating}_{\text{away\_team}} - \eta \times \text{error}$$

Where η is the learning rate (initially set to 0.01), and *error* is the difference between the actual and predicted goal difference. The home team's rating is adjusted upwards if the error is positive (i.e., the predicted difference was smaller than the actual), and the away team's rating is adjusted downwards by the same magnitude.

3.2.1 Centering the Ratings

After every iteration, the ratings are centered by subtracting the mean rating from all teams. This step ensures that the ratings remain balanced and centered around zero, preventing any drift in values over time.

3.2.2 Learning Rate Decay

The learning rate η decays over time to improve convergence stability. The formula for decaying the learning rate is:

$$\eta_t = \frac{\eta_0}{\sqrt{t}}$$

Where η_0 is the initial learning rate, and t is the current iteration number. By reducing the learning rate over time, we allow larger adjustments early in the training process when the ratings are far from optimal, and smaller, more precise adjustments later as the ratings converge.

3.2.3 Final Output

After iterating through all matches, the final team ratings are returned. These ratings reflect the relative strength of each team based on the match results. Higher ratings indicate stronger teams, while lower ratings indicate weaker teams.

3.3 Practical Application and Future Use

This SGD based algorithm can now be used to assign ratings to teams in future tournaments, provided we have match result data. The ratings generated by this model can be used in several ways:

- **Predicting future match outcomes**: The difference in ratings between two teams can be used to predict the likely goal difference in their future matches.
- **Assessing team performance over time**: Comparing team ratings across multiple seasons or tournaments provides insights into how teams improve or decline in performance.
- **Ranking teams**: These ratings can serve as a ranking system, where teams are ordered by their relative strengths based on match results.

In this specific case, the algorithm was applied to the 2014, 2018, and 2022 World Cup dataset. By running this code, I generated ratings for all the teams in the tournament, providing a coefficient that reflects their performance based on the goal differences in each match. I then compared the coefficent ratings assigned to each team and how they were generally viewed at the time (if the good teams had higher ratings and the bad teams had lower ratings) and it seemed to do a pretty good job, meaning it can be useful in future scenarios

4 Auto encoder Framework and Training Methodology

In Ghosh's paper, they investigate the behavior of a simplified single neuron auto encoder using both Gradient Descent (GD) and Stochastic Gradient Descent (SGD) training methods.[3] The auto encoder function is defined as $f : R^n \to R^n$, parameterized by a single weight vector $w \in R^n$, and using either a linear activation function $\varphi(z) = z$ or a ReLU activation function $\varphi(z) = \max(0, z)$.

The goal of the auto encoder is to minimize the reconstruction error for each input $x \in R^n$. The encoder projects the input onto the direction of the weight vector w, and the decoder reconstructs the input using the same weight vector. This process is referred to as weight tied autoencoding since the same weight vector is used for both encoding and decoding.

We aim to understand how training methods, specifically full batch GD and mini batch SGD, affect the convergence of the weight vector w, as well as the quality of the features learned. In particular, they focus on the role of batch size in influencing convergence properties.

4.1 Autoencoder Function

Hinton and Salakhutdinov explain how an auto encoder typically involves two parts [4]:
Encoder Compresses the input data into a lower dimensional representation. **Decoder** Reconstructs the original input from the lower dimensional encoded representation.

In this single neuron setting, the autoencoder encodes an input x by projecting it onto the weight vector w and then decodes the projection back into the input space by scaling the same vector w. The encoding is performed as:

$$\text{encoding}(x) = w^T x$$

The decoding reconstructs the input by scaling the weight vector w by the encoded value:

$$\text{reconstruction}(x) = (w^T x) w$$

The auto encoder aims to minimize the reconstruction error, which is defined as the squared difference between the input x and the reconstruction:

$$\ell(w; x) = \frac{1}{2} \|x - (w^T x) w\|^2$$

Given a dataset $D = \{a_1, a_2, \ldots, a_m\}$, where the data vectors are orthonormal and $m \leq n$, the average reconstruction error over the dataset is given by:

$$L(w; D) = \frac{1}{m} \sum_{i=1}^{m} \ell(w; a_i)$$

4.2 Training with Gradient Descent (GD)

In full batch gradient descent (GD), the model updates the weight vector w by computing the gradient of the loss function $L(w; D)$ over the entire dataset. This means that each weight update takes into account the reconstruction error for all data points simultaneously. [3]

The update rule in GD is:

$$w \leftarrow w - \eta \nabla L(w; D)$$

where η is the learning rate and $\nabla L(w; D)$ is the gradient of the loss with respect to w computed over the entire dataset.

Since the gradient is computed based on all data points, GD tends to produce stable updates but can be computationally expensive for large datasets. It also runs the risk of getting stuck in local minima in highly non convex optimization problems.

4.3 Training with Stochastic Gradient Descent (SGD)

In stochastic gradient descent (SGD), the model updates the weight vector more frequently by computing the gradient based on only a single data point (or a mini batch of data points) at each iteration. This makes the updates more noisy but allows for faster training, particularly for large datasets. [3]

The update rule in SGD is:

$$w \leftarrow w - \eta \nabla \ell(w; a_i)$$

where a_i is a single data point selected randomly from the dataset, and $\nabla \ell(w; a_i)$ is the gradient of the loss function for that specific data point. In mini batch SGD, a small subset (batch) of data points is used instead of a single data point.

By introducing randomness into the gradient calculation, SGD can escape local minima more easily and often converges faster than GD. However, this comes at the cost of increased variability in the updates, which may lead to overshooting or oscillation near the optimal solution.

4.4 The Role of Batch Size

The size of the batch plays a critical role in the behavior of SGD. A smaller batch size introduces more randomness into the updates, which can help the model explore the loss landscape more effectively. However, smaller batches also increase the variance in the updates, potentially making the training process less stable. Conversely, larger batch sizes produce more stable updates but may slow down the convergence. [3]

The paper explores how different batch sizes and training methods (GD vs. mini batch SGD) influence the quality of the learned features and the convergence properties of the autoencoder. Specifically, the authors focus on how cyclic mini batch SGD affects the convergence compared to full batch GD.

5 Elo Rating System

The Elo rating system, originally developed for ranking chess players, has been widely adopted across various sports, including soccer, for ranking teams based on their performance. Lasek and Gagolewski [8] use Elo ratings in their paper to provide a dynamic mechanism to update team or player ratings after every game based on match outcomes, thereby reflecting relative performance levels. This section summarizes the key mathematical principles behind Elo ratings, focusing on how they are computed and updated iteratively.

5.1 Expected Win Probability

The Elo system's core function is the calculation of the **expected win probability** for team i playing against team j. This probability is based on the difference between the ratings of the two teams. The expected win probability for team i is calculated using the following logistic function:

$$P(\text{win}_i) = \frac{1}{1 + 10^{\frac{(\text{rating}_j - \text{rating}_i)}{D}}}$$

Where:

- $P(\text{win}_i)$ is the expected probability of team i winning the match.
- rating_i and rating_j are the current Elo ratings of teams i and j, respectively.

- D is a scaling factor, typically set to 400 in the standard Elo system, which controls the sensitivity of the rating difference. For example, a difference of 400 rating points means that the higher rated team is 10 times more likely to win.

This logistic function ensures that the expected probability $P(\text{win}_i)$ ranges between 0 and 1, with equal ratings resulting in a 50% chance for each team. As the rating difference increases, the probability of the higher rated team winning increases, and the lower rated team's chances decrease.

5.2 Updating Elo Ratings

After the match, both teams' ratings are updated based on the actual outcome of the game compared to the expected outcome. The Elo rating system uses the following update rule for team i:

$$\text{new rating}_i = \text{rating}_i + K \times (\text{actual outcome}_i - P(\text{win}_i))$$

Where:

- K is the rating adjustment constant (commonly set to 32 for most sports, but adjustable depending on the sport's competitive balance).
- actual outcome$_i$ is 1 if team i wins, 0.5 for a draw, and 0 if team i loses.
- $P(\text{win}_i)$ is the expected win probability for team i, calculated using the logistic function above.

For team j, the opponent, the rating update is given by:

$$\text{new rating}_j = \text{rating}_j + K \times (\text{actual outcome}_j - P(\text{win}_j))$$

Since $\text{actual outcome}_j = 1 - \text{actual outcome}_i$, and $P(\text{win}_j) = 1 - P(\text{win}_i)$, the ratings for both teams are adjusted symmetrically, with the winner's rating increasing and the loser's rating decreasing.

5.2.1 Detailed Explanation of Rating Update

To better understand the rating update process, consider the case of a match where team i plays against team j. The formula for team i's new rating can be broken down into its components:

$$\text{rating}_i = \text{rating}_i + K \times (\text{actual outcome}_i - P(\text{win}_i))$$

$$\text{rating}_j = \text{rating}_j + K \times (\text{actual outcome}_j - P(\text{win}_j))$$

If team i wins ($\text{actual outcome}_i = 1$), the adjustment is based on how unexpected the win was. For example, if team i was expected to win with $P(\text{win}_i) = 0.8$, their rating will not increase much since their win was expected. However, if they were expected to lose ($P(\text{win}_i) = 0.2$), their rating will increase significantly, reflecting an unexpected win.

Similarly, if team i loses ($\text{actual outcome}_i = 0$), their rating decreases more if they were expected to win. This mechanism ensures that the Elo rating system accounts for both the match result and how surprising the outcome was.

5.3 Initialization of Elo Ratings

Teams typically start with the same initial rating, often set to 1500 for new teams, which represents a "neutral" skill level. As more matches are played, the Elo system dynamically adjusts the ratings based on performance, gradually reflecting each team's relative strength. For example, at the beginning of a tournament, all teams may have identical ratings, but as the tournament progresses, the ratings will shift based on wins, losses, and draws.

The initialization formula can be expressed as:

$$\text{rating}_{\text{initial}} = 1500$$

This provides a uniform starting point for all teams, allowing the Elo system to adjust ratings based on match outcomes.

5.4 Handling Draws in Elo Ratings

Draws are treated differently in the Elo system compared to wins and losses. When two teams draw, both teams' ratings are adjusted toward each other, meaning the higher rated team loses points, and the lower rated team gains points. The actual outcome for a draw is represented as 0.5. The update formula for draws is:

$$\text{new rating}_i = \text{rating}_i + K \times (0.5 - P(\text{win}_i))$$
$$\text{new rating}_j = \text{rating}_j + K \times (0.5 - P(\text{win}_j))$$

This formula ensures that if the higher rated team was expected to win and the match results in a draw, the higher rated team will lose points, and the lower rated team will gain points, reflecting the "unexpected" outcome.

5.5 Iterative Updates for Multiple Matches

Elo ratings are updated iteratively over time. For a dataset of matches, such as a soccer tournament, each match updates the ratings of the participating teams. The iterative process can be summarized as follows:

1. Retrieve the ratings for the two teams.
2. Compute the expected win probability for both teams using the logistic function.
3. Update the ratings based on the actual match outcome.
4. Use the updated ratings for future matches.

For example, after every match in the 2014 World Cup, the Elo ratings for the participating teams were adjusted according to the formulas above. As more matches are played, the ratings become a more accurate reflection of each team's performance.

5.6 Practical Application of Elo Ratings

The Elo rating system is particularly useful for ranking teams and predicting future match outcomes. The iterative nature of the system ensures that ratings dynamically reflect recent performance, and the logistic function models the relationship between rating differences and win probabilities.

The Elo system is widely used in soccer, chess, and other competitive sports because of its ability to handle wins, losses, and draws efficiently. It is flexible and can be adapted to different sports by adjusting the K value or the scaling factor D to account for the specific characteristics of the sport.

5.7 Conclusion

In conclusion, the Elo rating system provides a dynamic and interpretable method for ranking teams based on performance. By updating ratings after each match and using expected win probabilities to determine the adjustments, the Elo system continuously reflects the relative strength of teams. The mathematical foundations, including logistic regression and iterative updates, make Elo ratings an effective tool for sports analytics and forecasting.

6 Markov Chain Framework

Chapter 2 of Sokol's article [13] focuses on the application of a **Markov chain model** for ranking NCAA basketball teams. This model is based on a framework previously applied to NCAA football rankings. The key idea is to treat each team as a "state" in the Markov chain, with transitions between these states representing match outcomes. In the Markov chain model, each team is represented as a **state**. The transitions between these states simulate the decision making process of a hypothetical "voter," who initially believes one team is the best and then reevaluates their judgment based on the outcomes of games.

6.1 Markov Chain Definitions

A Markov chain consists of a collection of states and a set of transition probabilities P_{ij} that define the probability of moving from state i to state j. In the context of sports rankings, each team is represented by a state, and the probability P_{ij} represents the likelihood of transitioning from team i to team j, based on match results. For the transition matrix P in a Markov chain, each row sums to 1:

$$\sum_{j=1}^{N} P_{ij} = 1, \quad i = 1, \dots, N$$

where N is the total number of teams.

6.2 Conditions for a Unique Stationary Distribution

Ross [11] explains that a Markov chain is **irreducible** if it is possible to reach any state from any other state with a positive probability. This property is crucial in ensuring that the Markov chain is fully connected, meaning no subset of states is isolated. In mathematical terms, a Markov chain is irreducible if, for all states i and j, there exists a positive integer n such that:

$$P(X_n = j \mid X_0 = i) > 0.$$

This implies that it is possible to transition from state i to state j in n steps, guaranteeing that all states communicate with one another.

6.2.1 Aperiodicity

In addition to being irreducible, the Markov chain must also be **aperiodic** to ensure the existence of a **unique stationary distribution**. A state i in a Markov chain has a period d if d is the greatest common divisor (gcd) of all integers n for which $P(X_n = i \mid X_0 = i) > 0$. A chain is aperiodic if the period of every state is 1. Mathematically, this condition means:

$$\gcd\{n : P(X_n = i \mid X_0 = i) > 0\} = 1, \quad \forall i.$$

Equivalently, a Markov chain is aperiodic if there exists some n such that:

$$P(X_n = j \mid X_0 = i) > 0 \quad \text{for all states } i \text{ and } j.$$

Aperiodicity ensures that the chain does not exhibit cyclic behavior, allowing transitions to occur at irregular intervals rather than being constrained to fixed patterns.

6.2.2 Ergodicity and Unique Stationary Distribution

A Markov chain is said to be **ergodic** if it is both irreducible and aperiodic. These properties together guarantee that the chain has a **unique stationary distribution** π. The stationary distribution satisfies the following balance equation:

$$\pi P = \pi,$$

where P is the transition matrix of the chain. Additionally, π is a probability vector, meaning:

$$\sum_i \pi_i = 1 \quad \text{and } \pi_i \geq 0 \text{ for all } i.$$

The stationary distribution represents the long term behavior of the Markov chain, where the probabilities of being in each state remain constant over time. Starting from any initial distribution, the chain converges to π as the number of steps approaches infinity:

$$\lim_{n\to\infty} P^n = \mathbf{1}\pi,$$

where P^n is the n-step transition matrix and $\mathbf{1}$ is a vector of ones.

6.2.3 Implications of Ergodicity

The ergodic properties of a Markov chain ensure that:

- **Existence and Uniqueness**: The stationary distribution π exists and is unique.
- **Independence from Initial State**: The long term behavior of the chain does not depend on its initial state.
- **Convergence**: The chain converges to the stationary distribution as the number of steps increases.

Ergodicity is thus a crucial condition for the meaningful application of Markov chains in modeling long term behavior, as it ensures both stability and predictability of the system.

6.2.4 Conclusion

The combination of irreducibility and aperiodicity ensures that a Markov chain is ergodic, leading to the existence of a unique stationary distribution. This distribution serves as a fundamental tool for analyzing the long term dynamics of the system, providing a stable and consistent framework for probabilistic modeling.

6.3 States and Transition Probabilities

The transition between states (teams) is described using a **transition matrix** T, where each element t_{ij} represents the probability of transitioning from team i's state to team j's state. The general formula for the transition probabilities is given by:

$$t_{ij} = \frac{1}{N_i} \sum \left(p \cdot I_{ik} + (1-p) \cdot (1 - I_{ik})\right)$$

where:

- N_i is the total number of games played by team i.
- I_{ik} is an indicator variable, which equals 1 if team i wins game k, and 0 if team i loses.
- p is a probability factor that reflects the voter's confidence in the match outcome.

6.4 Steady State Probabilities

The **steady state probabilities** π represent the long term probabilities that a team is believed to be the best, based on the results of its games. The steady state vector π satisfies the following system of equations:

$$\pi_j = \sum_{i=1}^{N} \pi_i P_{ij}, \quad j = 1, \ldots, N$$

where:

- π is the vector of steady state probabilities for all teams.
- T is the transition matrix.

Equivalently, in matrix form, the steady state condition is:

$$\pi T = \pi$$

with the normalization constraint:

$$\sum_{j=1}^{N} \pi_j = 1$$

This system can be solved iteratively using methods such as the **power iteration method**:

$$\pi^{(n+1)} = \pi^{(n)} T$$

The team rankings are derived from these steady state probabilities: the team with the highest steady state probability is ranked first, and so on.

6.5 Expected Values with Markov Chains

In addition to ranking, Markov chains can estimate expected values for various quantities of interest. If $h(j)$ represents some function or performance metric for team j, the expected long term value of $h(X)$ is given by:

$$E[h(X)] = \sum_{j=1}^{N} h(j)\pi_j$$

where π_j represents the steady state probability of the chain being in state j.

6.6 Alternative Transition Probabilities

The transition probabilities can also be adjusted to account for additional factors like the **margin of victory** and **home court advantage**. These adjustments improve the accuracy of the rankings by incorporating more detailed game information.

6.6.1 Margin of Victory Adjustment

If team i wins by a large margin, the transition probability t_{ij} can increase more than for a narrow victory. This helps the model account for how decisively a team won a game.

6.6.2 Home Court Advantage Adjustment

Home court advantage can also be factored into the transition probabilities. The probability p may be adjusted when a team plays at home, reflecting the increased likelihood of winning when on their own court.

6.7 Calculating Team Rankings

The final team rankings are obtained by solving for the steady state probabilities π. These probabilities are computed iteratively until convergence to a steady state vector π that provides the final rankings for all teams.

6.8 Conclusion

The Markov chain model offers a robust method for ranking teams based on game results. By incorporating factors like the margin of victory and home court advantage, the model captures the relative strengths of teams in a more nuanced way than simple win/loss records. The final rankings are derived from the steady state probabilities, which reflect the teams' long term performance and likelihood of winning future games.

7 Transition Probabilities

In chapter 3 of Sokol's article [13] they begin to discuss the use of **logistic regression** to calculate transition probabilities between teams, which are used to construct the transition matrix in the Markov Chain model. These probabilities form the core of the ranking model, determining how likely one team is to be superior to another based on past game outcomes. The transition probabilities, denoted as H_x, estimate the probability that Team A is better than Team B given the result of a match between them. The match outcome is influenced by the **margin of victory**, which is incorporated into the logistic regression model.

The general logistic regression equation used to calculate transition probabilities is:

$$\ln\left(\frac{H_x}{1-H_x}\right) = a + b \cdot x$$

where:

- H_x is the probability of winning given a margin of victory x,
- a is the intercept parameter,
- b is the slope parameter that determines how much the margin of victory influences the probability.

7.0.1 Solving for the Transition Probability

To solve for H_x, the equation is rearranged to the following form:

$$H_x = \frac{1}{1 + e^{-(a+b\cdot x)}}$$

This equation represents the probability that Team A is superior to Team B given a margin of victory x. The larger the value of x, the higher the probability that Team A is stronger.

7.1 Adjusting for Home and Neutral Courts

The logistic regression model also accounts for whether a game is played at home or on a neutral court. Home court advantage is reflected by adjusting the probability estimate when a team plays at home.

7.1.1 Home Court Adjustment

The transition probability for home games is denoted as H_{xs}, which includes a home court advantage factor h. For neutral court games, the adjustment for home advantage is removed, and the probability is given by:

$$H_{xr} = H_{xs} + h$$

where h is the home court advantage adjustment. For games played on neutral courts, $h = 0$, meaning that the neutral court transition probability simplifies to the base model without the home court adjustment.

7.2 Logistic Regression Estimation

The logistic regression model is estimated using data from teams that played home and home series (where two teams play once on each other's home court). This data allows for the estimation of the parameters a and b in the logistic model.

The estimated logistic model is then used to calculate the likelihood of Team A winning future games against Team B, taking into account factors such as the margin of victory and whether the game is played on a neutral court.

7.3 Logistic Regression for Transition Matrix

Once the logistic regression parameters are estimated, the transition probabilities H_x are used to fill in the elements of the **transition matrix** T, where each element t_{ij} represents the probability of Team i transitioning to a superior state relative to Team j.

The transition matrix T is then used to simulate and predict outcomes for future match ups, particularly for neutral site games, which are common in tournaments like the NCAA basketball tournament.

7.4 Conclusion

Chapter 3 provides the mathematical foundation for calculating transition probabilities using logistic regression. These probabilities are essential for constructing the Markov chain's transition matrix, which ultimately determines team rankings. The use of margin of victory, home court advantage, and logistic regression ensures that the model captures the nuances of game outcomes beyond simple win/loss data.

8 PageRank Algorithm and Markov Chains

Kumar [6] explains that the PageRank algorithm, developed by Larry Page and Sergey Brin, ranks webpages based on their link structure and is one of the most prominent applications of Markov chains in web search. PageRank leverages the idea that a webpage is more important if many other pages link to it, especially if those linking pages are also authoritative. The algorithm models this behavior using a Markov chain, treating each webpage as a **state** and hyperlinks as **transitions** between states.

8.1 Web as a Markov Chain: Transition Matrix and Link Structure

The web can be represented as a directed graph, where each node (webpage) links to other nodes through directed edges (hyperlinks). This structure can be encoded in a **transition matrix** W for

N webpages, where each element W_{ij} represents the probability of moving from page i to page j:

$$W_{ij} = \begin{cases} \frac{1}{\text{outdegree}(i)} & \text{if there is a link from page } i \text{ to page } j \\ 0 & \text{if there is no link from page } i \text{ to page } j \end{cases}$$

where outdegree(i) denotes the number of links on page i. This transition matrix captures the likelihood of a user navigating between pages by following hyperlinks.

Handling Sinks and Dangling Nodes

A key issue in constructing the transition matrix W is dealing with **sink nodes** (or dangling nodes), which are pages with no outgoing links. These sinks disrupt the balance of the transition matrix because a random surfer reaching a sink would have no further transitions. To prevent this, PageRank addresses sinks by adding a "teleportation" mechanism, which is achieved through a damping factor.

8.2 Damping Factor and the Google Matrix

The **damping factor** α (usually set to 0.85) introduces a probability that a user will "teleport" to any page on the web, rather than following a link. This factor prevents the random surfer from getting trapped in sinks and ensures that the chain is both irreducible and aperiodic, satisfying the conditions for a unique steady state distribution.

The modified matrix, known as the **Google matrix** M, is defined as:

$$M = \alpha W + (1 - \alpha)R$$

where R is an $N \times N$ matrix with each entry equal to $\frac{1}{N}$, representing the probability of teleporting to any page uniformly. This construction ensures that the Markov chain represented by M is ergodic, meaning it has a unique stationary distribution that can be used for ranking.

8.3 Steady State Distribution and PageRank Scores

The **PageRank score** of each page is given by the steady state distribution of the Markov chain defined by M. Let π be the PageRank vector, where each entry π_i represents the probability of the random surfer being on page i. The PageRank vector satisfies:

$$\pi = \pi M$$

which is a left eigenvector of M corresponding to the eigenvalue 1. In component form, this is equivalent to:

$$\pi_j = \sum_{i=1}^{N} \pi_i M_{ij}, \quad j = 1, \ldots, N$$

subject to the normalization condition $\sum_{j=1}^{N} \pi_j = 1$. The value π_j reflects the long term proportion of time that a random surfer would expect to spend on page j.

8.4 Iterative Computation of PageRank

Due to the size of the web, calculating π directly is impractical for large networks. Instead, the PageRank vector π is computed iteratively using the **power iteration method**:

$$\pi^{(n+1)} = \pi^{(n)} M$$

Starting with an initial vector (often uniform), this iterative approach updates π until it converges to a stationary distribution. Convergence is generally achieved after a few hundred iterations, depending on the value of the damping factor α and the structure of M.

Convergence and Eigenvalue Analysis

The convergence rate of the power iteration method depends on the second largest eigenvalue of M. Since M is a stochastic matrix, its largest eigenvalue is 1. The spectral gap (difference between the largest and second largest eigenvalues) controls the speed of convergence. With a damping factor of $\alpha = 0.85$, the eigenvalue α ensures that M closely resembles W while remaining ergodic.

8.5 Interpretation of PageRank as a Markov Chain Problem

PageRank can be interpreted as a Markov chain model with the following properties:

- **Memoryless Property (Markov Property)**: The PageRank of each page depends only on the ranks of pages that link to it, regardless of how those pages were reached.
- **Stationary Distribution**: The stationary distribution of M reflects the long term importance of each page based on its connectivity within the web.
- **Ergodicity**: The damping factor ensures that M is irreducible and aperiodic, guaranteeing a unique steady state distribution.

8.6 Applications of PageRank Beyond Web Search

While originally developed for web search, PageRank has been adapted to various fields:

- **Social Network Analysis**: PageRank is used to identify influential nodes within social networks.
- **Scientific Citations**: In citation networks, PageRank ranks papers based on citation influence.
- **Recommendation Systems**: PageRank can be applied to rank items in recommendation engines by modeling user item interactions.

8.7 Importance of PageRank in Network Analysis

PageRank's use of Markov chain properties makes it robust and interpretable, providing insights into network structure that simple link counts cannot. By relying on a stationary distribution, PageRank captures not just the number of links to a page but the quality of those links, offering a nuanced measure of importance that is widely applicable.

8.8 Conclusion

The PageRank algorithm exemplifies the application of Markov chains in networked data. By modeling webpages as states in a Markov chain and using a transition matrix to represent link following probabilities, PageRank computes a stationary distribution that reflects each page's relative importance. The damping factor ensures ergodicity, allowing for reliable rankings essential to web search and adaptable to other networked domains.

9 Applying PageRank to College Sports: NCAA Basketball Case Study

Building upon the PageRank algorithm's mathematical foundation and its basis in Markov chains, we can explore how this algorithm is applied within the sports world, especially for ranking college basketball teams. Mathews [9] explores how PageRank offers a robust, data driven framework for assessing the relative importance of nodes within a network. In the context of sports, these nodes represent teams, and the edges represent outcomes of games between them. This chapter provides an in depth examination of how PageRank can rank college sports teams, focusing on its application in NCAA Division I men's basketball, before extending the discussion to similar applications in college soccer.

9.1 Modeling Teams and Games as a Directed Graph

In a sports network, each team is represented as a **node**, and each game outcome creates a **directed edge** from the losing team to the winning team. For example, if team A defeats team B, an edge is created from B to A, indicating that team B "transfers" credibility or rank to team A. This structure aligns closely with PageRank's original framework for webpages, where the rank of each page is bolstered by links from other reputable sources.

Each team's rank is influenced by the number and quality of teams it has defeated, mirroring the original PageRank concept of link based importance. However, in sports, additional factors like **margin of victory**, **venue** (home, away, or neutral), and **game recency** can be incorporated to create a more nuanced model.

9.2 Transition Matrix and Weight Adjustments for Sports Rankings

To represent this structure in a Markov chain, we define a transition matrix T where each element T_{ij} represents the probability of transitioning from team j to team i. The transition probabilities are determined by:

- **Game Outcome**: The transition from team j to i is based on team j's loss to team i.
- **Weighting by Margin of Victory**: Games won by large margins contribute more weight to the transition probability, indicating a more decisive victory.
- **Venue Adjustment**: Winning on the road is more challenging, so away game victories are given more weight than home game victories.
- **Game Recency**: More recent games are weighted more heavily than older games, as they better reflect current team performance.

These weights modify the transition matrix T, reflecting the importance of each game outcome. This matrix is then adjusted with a damping factor, similar to the Google matrix in PageRank.

9.3 Damping Factor and the Modified Transition Matrix

As in traditional PageRank, a **damping factor** α (usually set to 0.85) is used to prevent rank sinks (teams that are isolated or have limited interactions). This factor introduces a probability that a "random voter" could jump to any other team rather than strictly following game results. The modified transition matrix M is given by:

$$M = \alpha T + (1 - \alpha)R$$

where R is a matrix with each entry $R_{ij} = \frac{1}{N}$, representing the uniform probability of a jump to any team. The damping factor α stabilizes the rankings by ensuring all teams are accessible in the Markov chain, regardless of schedule structure or conference restrictions.

9.4 Computing Steady State Rankings

The PageRank scores for each team, representing their relative strength, are obtained by finding the **steady state distribution** of the matrix M. Let π be the ranking vector, where each entry π_i represents the PageRank score of team i. The vector π satisfies the equation:

$$\pi = \pi M$$

The scores are computed iteratively until convergence using the power iteration method, as discussed in the previous chapter. Teams with higher PageRank scores are ranked higher, as they have accumulated more credibility through game results against quality opponents.

9.5 Application to NCAA Basketball

In NCAA Division I men's basketball, this PageRank based approach has been used to objectively rank teams based on their season performance. The ranking system:

- Provides an unbiased, data driven approach to determining team strength.
- Incorporates all games played, adjusting for factors such as game location and margin of victory.
- Improves over traditional win loss records by emphasizing "quality" wins, where teams are rewarded for defeating highly ranked opponents.

This method has shown promise in evaluating teams for tournament selection and seeding, as it reduces subjective biases inherent in human voting or expert opinions.

9.6 Predictive Power of PageRank in NCAA Tournaments

In addition to ranking teams, PageRank has demonstrated utility in predicting game outcomes in NCAA tournaments. By assigning a probability of winning to the higher ranked team, this model can help forecast outcomes, with teams holding higher PageRank scores more likely to win. Although predicting sports outcomes remains inherently uncertain, PageRank based models have shown moderate success and can improve when further tuned for sports specific dynamics, such as injuries or roster changes.

9.7 Applying PageRank to NCAA Soccer

The NCAA college soccer structure closely mirrors that of NCAA basketball, with teams competing within conferences and a national tournament that includes both **conference champions** and **at large bids**. The tournament is structured as a 64 team single elimination competition, similar to the NCAA basketball "March Madness" tournament.

Given this similarity, the PageRank algorithm can be effectively applied to NCAA soccer rankings:

- **Conference Play**: Like basketball, teams accumulate rank based on both intra and inter conference play, making PageRank an adaptable model for both sports.
- **Tournament Selection**: PageRank can aid in selecting the best at large teams by objectively ranking teams based on season performance.
- **Predictive Analysis**: By using the same PageRank framework, soccer teams can be ranked and game outcomes forecasted, making this a versatile tool for both ranking and prediction across NCAA sports.

9.8 Conclusion

The PageRank algorithm provides an adaptable framework for ranking teams in college sports by leveraging game outcomes, incorporating factors like venue and margin of victory, and ensuring consistent rankings via the damping factor. While its application in college basketball has demonstrated the model's value in ranking and predicting team strength, NCAA soccer presents a similar structure where PageRank can be similarly beneficial. This approach not only enhances ranking accuracy but also offers a foundation for extending quantitative rankings and predictions across various collegiate sports.

10 Proof of Zero Sum Property in Elo Ratings

10.1 Preliminaries

Let R_A and R_B be the pre match Elo ratings of two teams A and B, respectively. After a match between these two teams, their ratings are updated to R'_A and R'_B. The update formulas for the ratings are:

$$R'_A = R_A + K \cdot M \cdot (S_A - E_A),$$
$$R'_B = R_B + K \cdot M \cdot (S_B - E_B),$$

where:

- S_A and S_B are the actual outcomes of the match ($S_A = 1, S_B = 0$ if A wins; $S_A = 0, S_B = 1$ if B wins; $S_A = S_B = 0.5$ for a draw),
- E_A and E_B are the expected outcomes:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}, \qquad E_B = 1 - E_A,$$

- K is the rating adjustment factor,
- M is the margin of victory multiplier.

10.2 Total Post Match Ratings

The total post match ratings for teams A and B are:

$$R'_A + R'_B = \big(R_A + K \cdot M \cdot (S_A - E_A)\big) + \big(R_B + K \cdot M \cdot (S_B - E_B)\big).$$

Combine terms:

$$R'_A + R'_B = (R_A + R_B) + K \cdot M \cdot \big((S_A - E_A) + (S_B - E_B)\big).$$

10.3 Simplify $(S_A - E_A) + (S_B - E_B)$

By the definition of match outcomes:

$$S_A + S_B = 1,$$

and by the definition of expected outcomes:

$$E_A + E_B = 1.$$

Thus:

$$(S_A - E_A) + (S_B - E_B) = (S_A + S_B) - (E_A + E_B) = 1 - 1 = 0.$$

Substitute this result back into the equation for $R'_A + R'_B$:

$$R'_A + R'_B = (R_A + R_B) + K \cdot M \cdot 0,$$
$$R'_A + R'_B = R_A + R_B.$$

Generalizing to N Teams

Let the pre match ratings of all N teams in the system be:

$$\text{Total Pre Match Ratings} = \sum_{i=1}^{N} R_i.$$

After processing a match between any two teams A and B, the sum of their ratings remains constant:

$$R'_A + R'_B = R_A + R_B.$$

This property holds for every match in the dataset. Therefore, across all matches:

$$\text{Total Post Match Ratings} = \sum_{i=1}^{N} R'_i.$$

By induction on the number of matches, the total sum of ratings is conserved:

$$\sum_{i=1}^{N} R'_i = \sum_{i=1}^{N} R_i.$$

10.4 Conclusion

The Elo rating model satisfies the zero sum property because the total ratings of all teams in the system remain constant before and after every match, regardless of the number of matches played or the outcomes.

10.5 Verification with Empirical Data

To confirm the proof in practice, the algorithm calculates the total sum of ratings before and after all matches. Computational results validate that:

$$\text{Initial Sum of Ratings} = \text{Final Sum of Ratings}.$$

This empirical evidence aligns with the mathematical proof, demonstrating that the Elo rating model is zero sum.

11 Tree Based Estimation and Ensemble Methods

11.1 Tree Based Estimation

Tree based models are non parametric methods that partition the feature space into a set of rectangles and then fit a simple model (often a constant) in each one. The foundation of this approach is the idea of recursively splitting the data to minimize classification error or some other impurity measure.

Optimal Splits in 1D. For one dimensional data, the optimal decision boundary minimizes the total misclassification rate. This can be efficiently computed by examining the $n-1$ *gaps* between adjacent sorted data points. The misclassification rate is constant between points, so evaluating the midpoint between each pair is sufficient. This is conceptually related to the maximal margin idea in support vector machines (SVMs).

Recursive Partitioning. A split at a point (e.g., $x = 47$) divides the data into two branches: $x < 47$ and $x \geq 47$. The algorithm recursively considers further splits within each branch. This results in a tree structure where internal nodes represent split conditions and leaves correspond to terminal classification or regression regions.

High Dimensional Challenges. In higher dimensions, exact computation becomes intractable due to exponential growth in the number of possible partitions. For example, in p dimensions, the brute force search scales as $\mathcal{O}(n^p)$, making it infeasible.

CART. Classification and Regression Trees (CART) reduce this complexity by restricting to axis aligned splits. At each node, the algorithm considers only splits along one feature at a time and chooses the best split based on a criterion such as Gini impurity or misclassification rate. This makes the method scalable and interpretable.

11.2 Evaluating and Enhancing Classifiers

Limitations of Misclassification Rate. Misclassification rate does not account for asymmetries in classification costs. For instance, classifying a positive instance as negative might incur a much larger penalty than the reverse. Thus, it is more appropriate to use the *expected loss* or *risk*:

$$r(T) = \sum_{j=1}^{k} P(A_j) \cdot r(A_j)$$

where A_j are the terminal nodes and $r(A_j)$ is the risk within node j, computed based on the loss function and posterior class probabilities.

Bayes Optimal Classifier. The optimal classifier under expected loss is the Bayes classifier, which assigns each observation to the class minimizing the expected loss:

$$\text{Classify as } j^* = \arg\min_j \sum_{l \neq j} L(j, l) \cdot P(l \mid x)$$

For symmetric 0–1 loss, this reduces to assigning the most probable class (MAP rule).

11.3 Ensemble Methods

Bagging. Bootstrap Aggregating (Bagging) is a technique to reduce variance by averaging over multiple models. The procedure:

1. Draw B bootstrap samples from the dataset.
2. Train a separate tree on each sample.
3. Aggregate predictions (average for regression, majority vote for classification).

11.3.1 Random Forests

[2] explains how Random Forests are an ensemble learning method that builds upon the idea of bagging by adding an additional layer of randomness to further reduce the variance of decision trees. While bagging constructs multiple trees on bootstrapped samples of the data, Random Forests also introduce randomness in the selection of features used to determine each split in a tree. This de correlation among the individual trees makes the overall ensemble more robust and less prone to overfitting.

Motivation. Single decision trees are powerful due to their interpret ability and ability to model non linear relationships. However, they tend to have high variance, small changes in the training data can lead to completely different trees. Bagging reduces this variance by averaging over multiple trees trained on different bootstrap samples. Still, if the trees are strongly correlated (due to dominant features), the gains from averaging may be limited. Random Forests mitigate this by forcing diversity among trees through randomized feature selection.

Training Procedure. The Random Forest algorithm works as follows:

1. For each of B iterations:
   (a) Draw a bootstrap sample of the training data.
   (b) Grow an unpruned decision tree from the sample:
      - At each node, instead of considering all p predictors, randomly select m features ($m < p$).
      - Choose the best split among these m features using a criterion such as Gini impurity (classification) or variance reduction (regression).
2. Aggregate predictions:
   - For regression: output the average prediction across all trees.
   - For classification: output the majority vote.

Hyperparameters. Key hyperparameters in Random Forests include:

- `n_trees` (B): number of trees to grow. More trees reduce variance but increase computation time.
- `max_features` (m): number of features considered at each split. Common defaults:
  - $\sqrt{p}$ for classification
  - $p/3$ for regression
- `min_samples_leaf`: minimum number of samples required at a leaf node.
- `max_depth`: maximum depth of the tree (optional to prevent overfitting).

Out of Bag (OOB) Error. Each tree is trained on a bootstrap sample, leaving approximately $1/3$ of the training data unused (out of bag). These OOB samples can be used to evaluate prediction performance:

- For each observation, average the predictions from trees that did *not* include it in their bootstrap sample.
- Compute prediction error on these OOB predictions.

OOB error provides an unbiased estimate of the generalization error, eliminating the need for a separate validation set or cross validation.

Variable Importance. Random Forests naturally offer variable importance measures:

- **Mean Decrease in Accuracy**: For each feature, permute its values in the OOB data and measure the increase in prediction error.
- **Mean Decrease in Gini**: Track the total decrease in Gini impurity (or variance for regression) that results from splits on a given feature, averaged over all trees.

These metrics help identify which features are most influential in the model's decisions.

Theoretical Properties. Random Forests are consistent under certain conditions: as the number of trees grows, the generalization error converges to a limit. While each tree may overfit, their average prediction stabilizes due to the law of large numbers. This aggregation ensures that Random Forests maintain low bias (due to deep trees) while drastically reducing variance (due to averaging and feature randomness).

Advantages.

- High predictive accuracy with minimal tuning.
- Robust to overfitting due to averaging.
- Invariant to monotonic transformations of features.
- Can handle both classification and regression tasks.
- Naturally accommodates missing data and unbalanced classes.

Limitations.

- Less interpretable than single decision trees.
- Computationally intensive for large datasets or high dimensional feature spaces.
- May not extrapolate well in regression (constant outputs outside training range).

12 KenPom and Strength of Schedule Rankings

12.1 Overview of KenPom Style Rankings

KenPom rankings, named after Ken Pomeroy, are a widely used statistical rating system in NCAA Division I men's basketball. The methodology adjusts team performance metrics by accounting for the quality of competition, game location, and tempo, resulting in more robust and predictive rankings than simple win loss records. At the core of KenPom's system are efficiency metrics such as adjusted offensive and defensive efficiency, but a critical component underlying team evaluation is **Strength of Schedule (SOS)**.

SoS metrics aim to quantify how difficult a team's schedule has been, serving as an adjustment layer in overall team evaluations. This becomes especially crucial in leagues or tournaments where teams do not face each other in a round robin format, making direct comparison via win percentages misleading.

12.2 NCAA's Official RPI System

[10] specifies how The Ratings Percentage Index (RPI) is one of the NCAA's earliest attempts to quantify team performance relative to schedule strength. Though it has been replaced in many sports by more advanced systems like the NCAA NET ranking in basketball, the RPI formula remains influential and has been used extensively in collegiate soccer, baseball, and volleyball.

The traditional RPI formula is given by:

$$\text{RPI} = 0.25 \cdot \text{WinPct} + 0.50 \cdot \text{OWP} + 0.25 \cdot \text{OOWP}$$

Where:

- **WinPct** is a team's raw win percentage (ties typically counted as half wins).
- **OWP** is the average win percentage of a team's opponents.
- **OOWP** is the average OWP of a team's opponents' opponents.

The weights—25%, 50%, 25%—are designed to emphasize the strength of competition over simple record alone. While the RPI is limited by its simplicity and lack of score margin consideration, it introduced the recursive evaluation of schedule difficulty that modern systems build upon.

12.3 Strength of Schedule via OWP and OOWP

A commonly used approach in KenPom style systems, especially for sports like soccer and basketball, is to define Strength of Schedule as a combination of:

- **Opponent Win Percentage (OWP)**: The average win percentage of a team's opponents.
- **Opponents' Opponent Win Percentage (OOPW)**: The average OWP of a team's opponents' opponents.

This two level structure captures both the direct quality of a team's competition and the broader competitive environment in which those teams played.

Opponent Win Percentage (OWP)

For a given team t that plays a set of opponents $\mathcal{O}_t$, the opponent win percentage is calculated as:

$$\text{OWP}_t = \frac{1}{|\mathcal{O}_t|} \sum_{o \in \mathcal{O}_t} \text{WinPct}_o$$

The win percentage of an opponent o is computed as:

$$\text{WinPct}_o = \frac{\text{Wins}_o + 0.5 \cdot \text{Ties}_o}{\text{TotalGames}_o}$$

Ties are weighted as half a win to reflect partial success, a practice commonly accepted in soccer rating systems.

Opponents' Opponent Win Percentage (OOPW)

[7] further refines the context in which each team has competed using OOPW, the model includes the average OWP of each opponent's opponents. For a team t, the OOPW is given by:

$$\text{OOPW}_t = \frac{1}{|\mathcal{O}_t|} \sum_{o \in \mathcal{O}_t} \text{OWP}_o$$

This adds a recursive layer, rewarding teams that face opponents with difficult schedules themselves.

Weighted Combination for Final SOS

The final Strength of Schedule score combines both metrics with empirically motivated weights:

$$\text{SOS}_t = 0.67 \cdot \text{OWP}_t + 0.33 \cdot \text{OOPW}_t$$

This 2/3 to 1/3 weighting prioritizes direct competition while still accounting for systemic strength in a team's network of opponents. The balance is consistent with methodologies found in NCAA selection frameworks and public sports analytics literature.

12.4 Importance in College Basketball and Soccer

Strength of Schedule rankings are particularly valuable in collegiate sports where:

- Teams play highly unbalanced schedules across conferences and regions.
- Not all opponents are created equal; margin of victory and win rate need contextualization.
- Ranking and seeding decisions (e.g., NCAA tournament selection) must be made fairly across disparate competitive environments.

In soccer, S)S is also important due to the relatively low number of games, the frequency of tied results, and the limited number of inter conference matchups. Systems that include OWP and OOPW have been shown to better differentiate between teams that accumulate wins against weak opponents and those that are competitive in stronger environments.

Summary. The KenPom style SOS formulation using a weighted combination of OWP and OOPW offers a principled and interpretable framework for evaluating schedule difficulty. Its recursive, network aware structure allows it to effectively account for the contextual strength of results, making it a foundational component in modern sports analytics for both basketball and soccer.

13 Motivation

Southern California Intercollegiate Athletic Conference (SCIAC) teams have become increasingly underrated in national NCAA Division III men's soccer rankings and NCAA tournament selections over the last several years. SCIAC teams, even after delivering strong performances in the postseason, have struggled to be granted at large bids to the NCAA tournament. Over the last four finished seasons, only the SCIAC tournament champion has earned a bid to the national tournament, via the conference's automatic bid. No SCIAC team has received an at large bid during this time period.

This is particularly noteworthy given the success SCIAC champions have experienced when afforded the opportunity. In three of the past four years, the SCIAC representative advanced to the Sweet 16 of the national tournament, each year falling by a score of 1–0 to a team that advanced to the national championship game. This trend strongly suggests that the perceived difference in quality between SCIAC teams and top ranked programs is smaller than current ranking systems reflect.

One classic instance occurred four years ago when Claremont Mudd Scripps (CMS) completed the regular season with a perfect record and was ranked in the national top 10. However, after a loss in the SCIAC tournament championship, CMS did not receive an at large bid, effectively ending its season despite its exemplary résumé. The scenario is a quintessential illustration of how structural ranking or selection biases can unfairly punish SCIAC teams disproportionately.

One likely reason for this underrating is the SCIAC's geographical isolation. Due to travel restrictions and limited opportunities, SCIAC teams prefer to play the majority of their games within the conference. As a result, they have few out of conference games during the preseason, which restricts their exposure to nationally competitive teams.

One can wonder at this point whether this problem is unique to soccer. After all, the other SCIAC sports, such as basketball or baseball, have the same geographic constraints but don't seem to suffer from the same degree of exclusion from national attention. The most significant difference is the academic calendar. Soccer is scheduled during the fall semester, which has no significant academic break. Unlike winter or spring sports, fall sports such as soccer don't enjoy the benefit of a built in opportunity to schedule non conference games.

For example, the Pomona Pitzer men's basketball team managed to squeeze in three non conference games during winter break this past year, while the baseball team had four non conference series at spring break. Academic breaks provide flexibility and travel options that significantly increase a team's schedule and national exposure. In contrast, soccer programs are attached to the academic schedule at the start of the semester through finals, and it becomes logistically awkward, and in some cases, impossible, to travel for out of region games once the season begins. Thus, soccer programs like those within the SCIAC are left with limited mechanisms to build a nationally competitive resume outside of their local area.

This isolation has algorithmic implications. Under ranking systems in sports networks such as Markov Chains or PageRank, where team value is influenced by the opponent strength and win likelihood transitive diffusion, competition within a majority within a closed network such as SCIAC produces a form of "rank containment." Because win probability does not propagate outward toward the national graph, the aggregated win strength among SCIAC teams gets cycled internally. Thus, these models may underestimate SCIAC teams not due to their inferior quality, but due to structural marginalization, restricting their inclusion in the wider competitive network.

This study attempts this hypothesis by implementing and comparing a variety of ranking models, such as Elo, Markov Chains, PageRank, and Strength of Schedule measurements, on actual and simulated data sets to attempt to measure how under valued SCIAC teams may be and to what degree the adjustments can be made in an attempt to better reflect their true competitive strength.

14 Model Implementation and SCIAC Case Study

14.1 Custom Markov Chain Ranking Model with Margin Adjustment

Building upon the mathematical foundations discussed in Section 6 (Markov Chain Framework) and Section 7 (Transition Probabilities), I implemented a Markov Chain based ranking algorithm designed to incorporate both the number of games played between teams and the margin of victory. The core objective was to develop a more context sensitive team rating system that could identify latent bias in current rankings, particularly the consistent underrepresentation of SCIAC teams in NCAA tournament selection.

Motivation and Structural Bias Against SCIAC Teams

As detailed in Section 1, over the past four years, SCIAC teams have received only one automatic bid to the NCAA tournament each year, with no at large selections despite consistent postseason success. Three of those four SCIAC champions advanced to the Sweet 16 and narrowly lost to the eventual national finalists by a 1–0 scoreline. This pattern reveals that SCIAC teams are significantly undervalued in traditional rankings. A primary driver of this discrepancy is geographic isolation, which limits the number of non conference games and therefore restricts their exposure in network based models such as PageRank and Markov Chains.

Transition Matrix Construction with Laplacian Smoothing

To address this, I implemented a customized transition matrix P where entry P_{ij} represents the weighted likelihood that team i is stronger than team j, derived from match results. The construction process uses the following method:

1. For each pair of teams i and j, extract all games played.
2. For each match, calculate a smoothed win probability p using the formula:
$$p = \frac{\text{score}_i + \alpha}{\text{score}_i + \text{score}_j + 2\alpha}$$
where α is a smoothing constant and scores are adjusted for home advantage.
3. Average probabilities over all head to head matches and scale by the number of games:
$$P_{ij} = (\#\text{games}_{ij}) \cdot \bar{p}_{ij}$$
4. Normalize rows of P so each row sums to 1, ensuring a valid stochastic matrix.

Extracting Rankings via Stationary Distribution

After constructing the transition matrix P, I computed its stationary distribution π by repeatedly multiplying P by itself:
$$\pi = \lim_{k\to\infty} \mathbf{1}^\top P^k$$
This steady state vector represents the long run proportion of time a random walker would spend at each team node, thereby forming the team rankings. This process was discussed in mathematical detail in Sections 6.2 and 6.3.

Empirical Findings and Validation

When applied to the full 2023 NCAA Division III men's soccer dataset, the model revealed stark discrepancies compared to official rankings. For instance, Occidental College, who advanced to the Sweet 16 and ended the season ranked 14th nationally, was ranked outside the top 100 in my Markov Chain model. This discrepancy stems from the SCIAC's low connectivity to the broader national schedule, leading to a recycling of win probabilities within the conference (see Section 1 for full justification).

This result not only demonstrates the effectiveness of my model in identifying structural ranking biases, but also highlights how network based models, while powerful, must be adjusted or augmented when applied to geographically siloed leagues.

Visualizing Team Connectivity via the Transition Matrix

To better understand the structural topology of inter team connectivity, I visualized the constructed transition matrix as a directed graph, where each node represents a team and each edge represents a transition probability derived from head to head matches. The resulting network graph, shown in Figure 1, clearly illustrates the overall interconnectedness among teams across the NCAA Division III landscape.

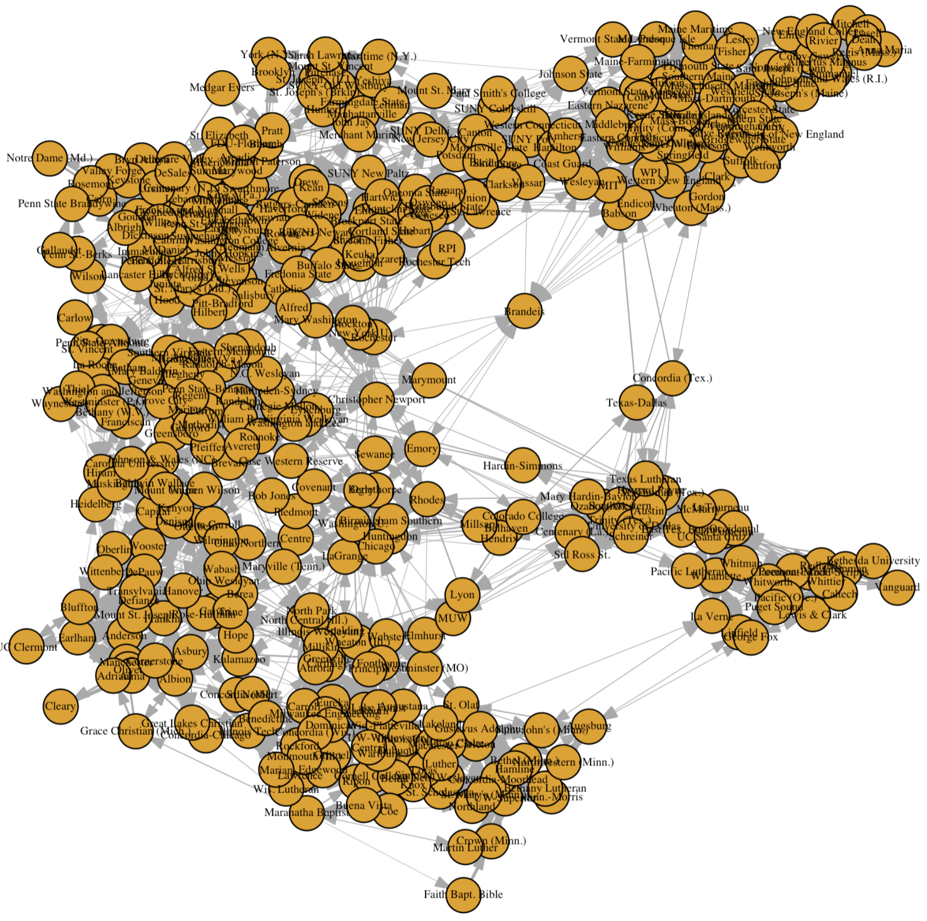


Figure 1: Visualization of the transition matrix as a directed network graph. Nodes represent teams, and directed edges represent weighted transition probabilities based on match outcomes. Node size indicates relative degree centrality.

What becomes immediately apparent from the visualization is the tight clustering of teams based on regional play. Most conferences are densely interconnected, forming large clusters in the network, while certain groups, particularly SCIAC teams located in the lower right region of the graph, are clearly more isolated.

This spatial isolation in the graph reflects the reality of SCIAC's limited non conference play due to geographic constraints. As a result, transition probabilities derived from SCIAC teams predominantly circulate within their own subgraph. In network terms, this causes the win probability to "recycle" internally rather than propagate across the full competitive landscape, leading to underestimation in PageRank and Markov based ranking systems. This visual confirms the theoretical limitations discussed earlier in Section 6 and 7 and motivates the need for adjusted or hybrid ranking models that can account for conference isolation.

14.2 Investigating Structural Isolation Using Centrality Measures

To further investigate whether the SCIAC conference's geographic isolation had a measurable impact on team valuations within network based ranking models, I computed **PageRank centrality** and **betweenness centrality** scores for all teams using the adjusted Markov chain transition matrix constructed earlier (see Sections 6 and 7).

This was accomplished by treating the transition matrix as a weighted, directed graph, where each node represents a team and edges encode probabilistic transitions derived from match results.

- **PageRank centrality** reflects the long term probability of arriving at a team in a random walk on the game result graph, and is closely tied to Markov chain stationary distributions.
- **Betweenness centrality** quantifies the extent to which a team lies on paths between other teams, indicating structural connectedness or gatekeeping roles in the network.

After calculating these measures, each team was labeled as either `SCIAC` or `Non SCIAC`, and distributions were compared using boxplots. Figure 2 shows the result of comparing the **PageRank centrality** values for SCIAC vs. non SCIAC teams.

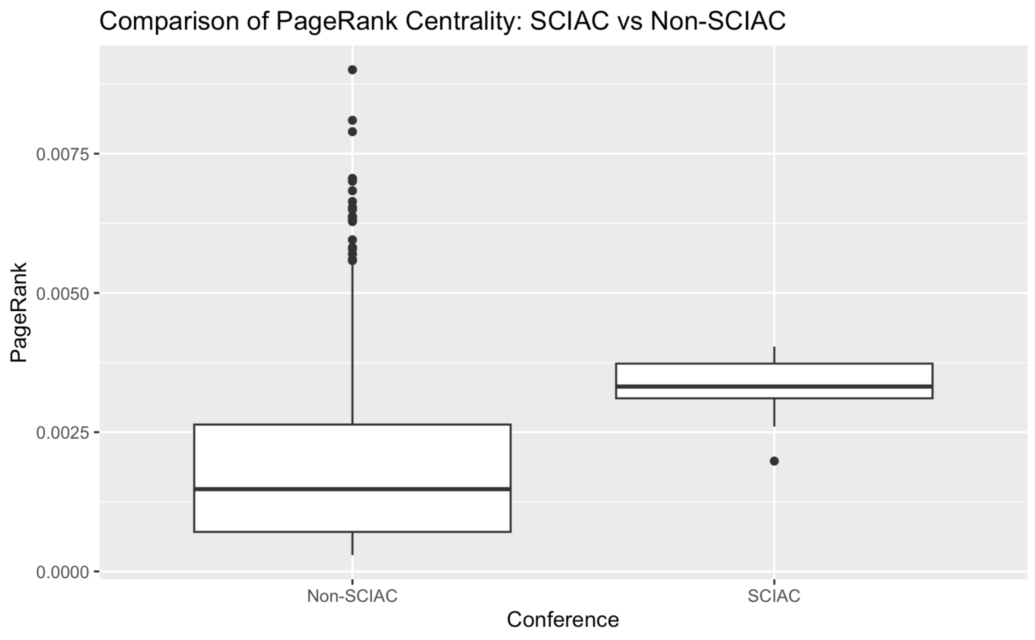


Figure 2: Comparison of PageRank centrality between SCIAC and non SCIAC teams. SCIAC teams, shown on the right, have notably less dispersion and lower median values than many non SCIAC teams, suggesting reduced visibility or impact in national win probability flows.

As illustrated, SCIAC teams display a much narrower and more concentrated distribution of PageRank scores, while non SCIAC teams are far more dispersed with many high outliers. This supports the hypothesis that SCIAC teams are structurally isolated in the game result network and therefore appear weaker than they are when evaluated by strength of schedule weighted metrics.

Such models, particularly those derived from transition based frameworks like PageRank or Markov Chains, are heavily dependent on interconnectivity. If win probabilities do not leave the conference, the teams' rankings remain trapped in a local subnetwork, reinforcing their lower centrality scores regardless of match performance. This finding further validates concerns that SCIAC teams are systematically undervalued due to geography induced network fragmentation.

14.3 Simulating Dominance: How Strength of Schedule Undervalues SCIAC

To further test whether SCIAC teams are systematically underrated by ranking systems that rely heavily on strength of schedule (SOS), I constructed a simulated dataset designed to reflect a world in which SCIAC teams are clearly the strongest in the country. This simulation allowed me to evaluate whether common strength based metrics would still underestimate these teams due to structural isolation.

In the simulated dataset, every SCIAC team was modeled as having significantly stronger scoring ability than any other team in the country. I applied a Poisson simulation where SCIAC teams were assigned an average goal scoring rate (λ) of 2.5 goals per match, while all other teams were assigned a scoring rate of only 1.0. This effectively created a controlled environment in which SCIAC teams dominated opponents with higher goal margins across the board. In theory, this should result in SCIAC teams being recognized as the strongest in any rating system that accurately reflects performance.

After generating simulated match results under these assumptions, I calculated the KenPom style Strength of Schedule (SOS) for all teams, just as I did in Section 9, using the weighted combination of opponent win percentage (OWP) and opponents' OWP (OOPW). I then compared the average SOS of SCIAC teams in the simulated dataset to their SOS in the real, historical dataset.

The results were revealing. In the simulated dataset, where SCIAC dominance was artificially ensured, the average KenPom SOS for SCIAC teams was: .528
By contrast, in the actual real world data, the average SOS for SCIAC teams was only: 0.51

For context, the highest SOS in the country was around .69 and the lowest around .22 so its clear that while there was an improvement, it was very minimal and nowhere near the jump we would expect from a conference's average SOS if they were objectively the best teams in the country.

While the simulated value is higher, the gap between the two is surprisingly narrow considering the extreme scoring advantage given to SCIAC teams in the simulation. Despite dominating nearly every game in the model, their calculated strength of schedule barely budged.

This reinforces the central issue: SOS based metrics, which dominate college soccer rankings (e.g., NCAA selection criteria, Massey Ratings, and RPI variants), are inherently constrained by schedule structure. Since SCIAC teams are geographically isolated on the West Coast, they play fewer interregional games, especially against the traditionally dominant teams from the Midwest and Northeast. Consequently, their opponents tend to have weaker win percentages, which in turn drags down SCIAC teams' OWP and OOPW even if they consistently win by large margins.

In short, this simulation demonstrates that even when SCIAC teams are hypothetically the best in the country, the structure of their schedule suppresses their perceived strength when viewed through the lens of SOS. This helps explain persistent underrating in national rankings, particularly in systems that fail to fully adjust for geographic and structural limitations in match scheduling.

15 Introducing Elo Ratings to Address SOS Limitations

After identifying that Strength of Schedule (SOS) alone cannot fairly evaluate teams from geographically isolated conferences like the SCIAC, I realized that a ranking model that incorporates SOS but is not entirely dependent on it would be far more appropriate. To address this, I began implementing and testing a series of alternative ranking systems. The first of these was the Elo rating system, a widely used method in chess, international soccer, and other competitive domains. While the mathematical theory behind Elo ratings is outlined in Section 5, this section focuses on my specific implementation and motivations.

15.1 My Elo Rating Framework for College Soccer

The goal was to build a dynamic rating system that updates over time based on match outcomes while carrying forward the context of past performance. To do this, I initialized Elo ratings not uniformly, but rather based on how each team finished in the previous year's NCAA Division III tournament. Teams that went deeper in the tournament were awarded higher starting ratings. This reflects how real world rankings tend to begin the season: with prior performance serving as the baseline until new results accumulate. Section 5 goes over the math and theory involved in this ranking system and why its an effective method used.

15.1.1 How Ratings were Determined

For example, the reigning national champion was given an initial rating of 1675, while non tournament teams started at 1500. This gave strong returning teams an early season advantage, but also set up a mechanism for their ratings to adjust downward if they underperformed.

Each match in the schedule was processed sequentially. For every game, I calculated the expected win probability using the difference in Elo ratings, adjusted by a home field advantage of +50 points for the home team. The actual result of the game was then compared to this expected outcome: - A win awarded 1 point, - A draw awarded 0.5, - A loss awarded 0.

The difference between actual and expected results determined the size and direction of the rating update. Additionally, I included a margin of victory multiplier to weight lopsided results more heavily, rewarding dominant performances more than narrow wins. This was governed by a tunable parameter β, which I set to 0.5. Larger goal margins led to larger rating adjustments, while tighter games produced smaller changes.

A zero sum constraint was applied to ensure that after every match, the total number of rating points in the system remained the same. This property reflects the principle that when one team rises in the rankings, another must fall. After all matches were processed, the final Elo ratings provided a ranked list of teams based on their results, strength of opposition, game locations, and score differentials.

15.1.2 Strengths and Weaknesses of My Model

This system allowed strong teams in less competitive conferences to accumulate rating points if they consistently won and did so convincingly. Because ratings transfer directly from opponent to opponent, this approach naturally accounts for a chain of performance over time. Even a team that starts with a lower rating can climb the rankings by defeating higher rated opponents, especially on the road or by large margins.

Compared to systems that rely predominantly on strength of schedule, the Elo model offers key advantages. It updates continuously with every result, rewards consistent winning regardless of opponent reputation, and naturally incorporates context like game location and score margin. However, it is not without limitations. Elo ratings can be sensitive to early season imbalances in initial ratings and may lag in recognizing sudden improvements or declines. Despite these trade offs, Elo served as another important option for ranking teams alongside static SOS based systems by providing another important way of ranking teams that takes into account actual performance over time. It remains an essential framework worth analyzing alongside strength of schedule based models.

16 Bayesian Markov Chain Rankings with Elo Informed Priors

After evaluating the Elo model and recognizing its strengths and limitations, I developed a new ranking system that builds on the structure of a Markov Chain but incorporates Bayesian priors.

This approach addresses a key limitation in both SOS based and Elo based systems: the former often neglects outcome information beyond opponent strength, while the latter relies heavily on match by match updates and is sensitive to initial ratings. The Bayesian Markov Chain model provides a complementary method by integrating prior information and stabilizing estimates for teams with limited data.

The core idea behind this model is to estimate how likely each team is to "transfer strength" to other teams based on actual game outcomes while still incorporating prior beliefs informed by last season's performance. Rather than updating ratings sequentially like the Elo model, the Bayesian Markov Chain approach builds a global transition matrix of pairwise win probabilities between teams, normalized into a Markov Chain, and then derives a stationary distribution to determine rankings.

16.1 My Model

To begin, I initialized prior ratings using the same framework from the Elo section. Tournament success from the previous season was converted into a tiered scale of Elo like scores, ranging from 1675 for the national champion to 1500 for non tournament teams. These ratings served as Bayesian priors to adjust the win/loss probabilities for the current season.

The key computation lies in the construction of the Bayesian transition matrix. For each match, I updated the win probabilities between the two teams using the Beta distribution framework, which naturally accommodates prior beliefs. Specifically:

For every team, I scaled its Elo based prior into an equivalent number of pseudo wins and pseudo losses, forming the parameters of a Beta distribution. For each game outcome, the result (win/loss/tie) updated the Beta parameters accordingly. The updated win probability from team A to team B became one entry in the transition matrix. Ties were handled symmetrically by awarding half a win to both teams.

This process produced a complete matrix where each row corresponds to a team's probabilities of beating every other team, incorporating both actual match outcomes and prior beliefs. I then normalized each row to ensure it sums to 1, turning the matrix into a valid stochastic matrix suitable for a Markov Chain.

To derive the rankings from this structure, I computed the stationary distribution of the transpose of the transition matrix. The stationary vector provides long run probabilities of being at a given team if we were to simulate an infinite random walk through this win probability graph. Higher stationary probabilities indicate stronger teams, as they receive more strength from beating others (and being beaten less frequently).

16.2 Differences from Elo Model

Importantly, this model differs from the Elo system in several ways:

- It is not iterative or sequential. All matches are considered simultaneously. - It incorporates prior beliefs probabilistically through the Beta distribution, rather than initializing and updating scores manually. - It handles noise more gracefully, especially for teams with few games.

In this way, the Bayesian Markov Chain model offers a more holistic picture of team performance, accounting for opponent strength, match results, and tournament pedigree all in a unified, global framework. It complements the Elo model by emphasizing cumulative relational performance rather than game by game momentum, and provides another credible method for rating teams in a way that blends prior information with empirical data.

17 Ensembling Multiple Models for a Holistic Ranking

Having compared individual models such as the Elo rating system and the Bayesian Markov Chain method, I realized that there was a single thread that ran through them all: they were all unique with varying strengths and weaknesses. Whereas Elo did a great job of maintaining game by game momentum and penalizing poor showings against strong opponents, the system was likely sensitive to opening season ratings and did not integrate naturally the big picture network of games played within teams. Bayesian Markov supplemented structure, smoothing, and transition based fairness without the temporal fidelity and momentum awareness of Elo.

No single model generated a perfect estimation of team strength. All methods excelled under certain conditions but failed under others. This observation led me to wonder about the ranking challenge as not nearly as much a one true answer problem, but rather as more of a problem that is voting or consensus type. With that, I then went about treating all models as a "coach's poll", an independent ranking system that represents one subjective view of team performance.

With this definition, I opted to take a number of different models and weight them based on how accurately they were able to forecast the teams that would ultimately end up with a top 25 national ranking. The more accurate the model was at forecasting top teams, the more weight its forecast would receive in the ultimate rankings.

The two original contributors to this set were the Elo model and the Bayesian Markov Chain model. I employed these as two different voters in the ensemble setup. Other than these, I included my original Markov Chain model, which predicts teams based on transitions between nodes (teams) using game outcomes.

For transition based analysis extension, I created a set of variants for the Markov Chain model. All of these used a different weighting methodology to determine probabilities for team to team transition, such as soccer specific nuances like margin of score, Low Favor Ties, SOS KenPom, and PageRank based weight. Effectively, I constructed various Markov models, each with a distinct perception of how wins and losses affect perceived strength. Similar to the Elo and Bayesian models, each of these Markov variations was an individual "coach" in the poll.

The ultimate ranking thus represented a consensus rank system: a weighted average of several expert judgments, each model casting its ballot on the basis of its established capacity to rank the top performing teams. This approach allowed me to tap the strengths of different methods while reducing their respective limitations to a finished and interpretable ultimate ranking system.

17.1 Standard Filtered Markov Chain Model

To create a foundational version of a Markov Chain ranking system, I began by filtering out teams that had not played enough games to warrant a meaningful analysis. Specifically, I required teams to have played at least four matches, ensuring that each row in the transition matrix had sufficient data to reflect a team's performance. This also allowed me to exclude teams that only played NAIA or JUCO opponents in preseason exhibitions, games that appear in the dataset but are not considered in official NCAA Division III rankings. Including those games would introduce noise and unfairly skew the transition probabilities, so I removed them from consideration to align with how actual rankings are calculated.

The core of this model is the transition matrix T, described in section 6, where each entry $T_{i,j}$ represents the weighted probability that team i would "transition" to team j, or in the context of rankings, that team j is stronger than team i. To calculate these probabilities, I aggregated all head to head matches between each pair of qualified teams.

For each match, I computed a probability of transition based on the score margin, adjusted with a Laplace smoothing constant to avoid zero division or extreme values. The weighting of match results was then amplified using a margin sensitive scaling factor $\beta = 1.2$. This meant that larger winning margins contributed more heavily to the perceived strength difference between two teams. Additionally, I applied separate multipliers for home and away wins to reflect the greater difficulty and informational value of away victories. Draws were treated as balanced results and downweighted relative to wins.

These match level probabilities were aggregated to compute a single average probability p_{ij} for each team pair, where:

- $T_{i,j} = p_{ij}$
- $T_{j,i} = 1 - p_{ij}$

In cases where two teams had not played each other, I estimated transition probabilities using a derived "team strength" metric that accumulated weighted outcomes from their existing matches. These derived probabilities were used to fill in missing entries of the transition matrix. Once every cell in the matrix was filled, I normalized each row so the transition matrix satisfied the stochastic property (each row sums to 1).

To ensure ergodicity and convergence of the Markov Chain, I applied a damping factor of 0.85. This created a small but nonzero probability of jumping from any team to any other, guaranteeing that the matrix had a unique stationary distribution.

Finally, I computed the stationary distribution of the transposed transition matrix, using the method described in section 6. This stationary vector reflects the long run proportion of time a random walker would spend on each team, effectively serving as that team's score or ranking. Teams with high stationary probabilities are those that consistently win direct matchups or beat teams that themselves perform well in the overall network of games.

This filtered Markov model thus balances real game outcomes with network effects, rewarding teams not just for winning, but for doing so decisively and against strong competition, while still producing a probabilistically grounded and globally connected ranking.

17.2 PageRank Inspired Markov Chain Model

This model builds on the Markov Chain framework described in section 6, but introduces a PageRank inspired adjustment to how transitions between teams are weighted. In contrast to the filtered Markov Chain model, which relied heavily on score margins and team strength estimates, this version implements a refined transition matrix that better emphasizes the contextual significance of wins, losses, and draws by applying tunable weightings and smoothing techniques.

To begin, I constructed a transition matrix where each row corresponds to a team, and each entry $T_{i,j}$ represents the relative likelihood of a transition from team i to team j based on observed match results. Rather than assuming symmetric influence from all games, I differentiated outcomes with outcome dependent multipliers:

- Home wins contributed a slightly reduced weight ($0.9 \cdot \text{margin}^{\beta}$) to reflect the expectation of winning at home.
- Away wins were rewarded with a slightly higher weight ($1.1 \cdot \text{margin}^{\beta}$) to emphasize the challenge of winning on the road.
- Draws assigned a symmetric weight of $0.5 \cdot \text{margin}^{\beta}$ to both teams.

Here, $\beta = 0.5$ controlled the influence of score margins, giving more credit to teams that win convincingly.

Once all match based weights were aggregated, each row in the transition matrix was normalized so that the total probability mass summed to one. To handle potential rows with zero total weight (e.g., for teams with no valid match data), I assigned a uniform distribution across all teams for that row. This ensured that the matrix remained stochastic.

A key innovation in this model was the use of Laplacian smoothing: a small constant $\alpha = 0.01$ was added to all entries of the matrix before renormalization. This served to eliminate any potential zero entries and further stabilize the model in the presence of sparse data.

To guarantee ergodicity, I applied a damping factor $d = 0.85$, blending each row of the matrix with a uniform probability vector. This allowed the Markov Chain to converge to a unique stationary distribution, regardless of initial team or data sparsity.

As described in section 6, I computed the stationary distribution of the transposed transition matrix using the dominant left eigenvector. This stationary vector represents the long run visitation frequency of each team under this probabilistic system of transitions. Teams with high ratings were those that consistently collected results against other strong teams, particularly when doing so away from home.

Compared to the filtered Markov model, this approach favored a more holistic and robust transition mechanism, adjusting for both game context and structural stability. Its incorporation of PageRank principles made it particularly well suited to ranking systems with uneven match distributions or noisy results.

17.3 Win Margin Weighted Markov Chain Model

This model modifies the standard Markov Chain approach by placing additional emphasis on the margin of victory between teams. While the traditional filtered Markov model (discussed in section 6) incorporated outcome based weights and strength estimates for unplayed matches, this version pushes further by allowing dominant wins to exert more influence over the final rankings.

I began by generating a square transition matrix where each entry $T_{i,j}$ represents the weighted likelihood of team i transitioning to team j in the network of results. These weights were not only based on win/loss outcomes, but also scaled by the number of times two teams faced each other and how decisive those results were.

For every match between two teams, I computed a probability that incorporated:

- The score margin: larger margins indicated more dominant wins.
- A Laplace smoothing parameter $\alpha = 0.01$ to stabilize probability estimates.
- Home vs. away context to adjust the base win probability.

These probabilities were calculated for every head to head matchup and then scaled by the number of times the two teams played each other. This ensured that the effect of multiple dominant performances would be accumulated. The values were then normalized across each row to create a proper stochastic matrix.

Unlike previous models, I did not explicitly apply a damping factor here. Instead, convergence to the stationary distribution was achieved through iterative matrix multiplication, simulating the behavior of a long run random walker through the transition network. After sufficiently many iterations, the row vectors of the transition matrix converged to the same distribution, which was then used to rank teams.

This win margin weighted model differs from the filtered model in that it prioritizes not only whether a team won, but how convincingly they did so. By aggregating and averaging performance over multiple games and emphasizing decisive results, this approach allowed consistently dominant teams to rise more easily in the rankings.

As before, a final ranking was extracted from the stationary distribution, where higher probabilities corresponded to stronger overall performance in the network of results. The use of a margin sensitive model captures an important layer of performance detail that other models may overlook when solely evaluating win/loss records or network connectivity.

17.4 Strength of Schedule KenPom Adjusted Markov Model

Although the Markov Chain framework inherently incorporates strength of schedule effects by virtue of the way transitions propagate through a network of games, in this model I made the role of schedule strength even more explicit. I did this by directly incorporating each team's KenPom style strength of schedule (SOS) rating into the transition probabilities between teams.

To begin, I computed KenPom style SOS scores for every team in the dataset, using the method described in section 5.2. These values were then normalized to lie between 0 and 1 to serve as weighting factors for match outcomes. This ensured that results against teams with stronger schedules were weighted more heavily in the final rankings.

The transition matrix was constructed such that the edge weight from one team to another depended not only on the game outcome and score margin (scaled by a margin impact factor $\beta = 1.2$), but also on the opponent's KenPom SOS score. For example, if team A beat team B by a substantial margin and team B had a high SOS, then the transition probability assigned to team A being stronger was heavily reinforced. Conversely, wins against weaker opponents were down weighted accordingly.

More specifically:

- For each game, I calculated the win margin (plus one to avoid zeros), and scaled it using β.
- The impact of each win or loss was weighted by the opponent's normalized SOS score, with a small constant α added for numerical stability.
- Ties were assigned half weighted transitions in both directions.

Once the raw transition matrix was built, I normalized each row so that it summed to one, ensuring the matrix satisfied the stochastic constraint. I then applied a damping factor of 0.85 to guarantee ergodicity, allowing the stationary distribution to converge.

The final rankings were determined by computing the stationary distribution of the resulting transition matrix, as described in section 6. This distribution reflects the long run probability of a random walker landing on each team, with stronger teams absorbing more of the overall probability mass.

This SOS adjusted model stands out from the others by explicitly formalizing the notion that not all wins or losses carry equal informational value. It rewards teams not just for winning, but for doing so against high caliber opposition, thereby producing rankings that are highly sensitive to the underlying strength of a team's schedule.

17.5 Low Favor Tie Adjustment

This final Markov Chain model builds on the base transition framework by rethinking how ties are interpreted in the ranking process. While previous models either treated ties as neutral outcomes (50/50 splits) or downweighted them slightly, this version introduced a tie handling mechanism that assigns additional weight to the lower ranked team in each drawn match. The goal was to reflect the real world intuition that when a weaker team ties with a stronger one, the result should speak more positively about the underdog.

As outlined in section 6, the model starts by constructing a transition matrix T where each element $T_{i,j}$ estimates the probability of transitioning from team i to team j based on game outcomes. All matches between each pair of teams were extracted from the full dataset, and a probability of transition was calculated for each individual game using Laplace smoothing and a margin based weighting factor $\beta = 1.2$.

However, the key innovation in this model lies in the tie adjustment. When a draw occurred, instead of defaulting to a 0.5-0.5 probability assignment, I shifted the balance slightly in favor of the team that entered the match as the perceived underdog. This was implemented by allocating

60% of the transition probability to the lower ranked team. This small but meaningful shift better captured the intuition that a draw can serve as a statement result for the less favored side.

The remaining steps mirrored the traditional Markov Chain workflow. The raw transition matrix was row normalized to ensure each row summed to 1, then adjusted using a damping factor of 0.85 to guarantee ergodicity. I then computed the stationary distribution of the transposed matrix, yielding a vector of long run visit probabilities that served as the final team rankings.

This low favor tie model offers a more nuanced interpretation of parity outcomes than previous approaches. By amplifying the implications of strong performances by weaker teams, it helps surface upsets and resilient performances that may otherwise go underappreciated in systems that treat ties too symmetrically.

18 Constructing the Final Composite Ranking Model

After developing several distinct ranking models, each with its own approach to capturing team strength based on game outcomes, score margins, connectivity, and schedule strength, I aimed to synthesize these models into a single, final ranking that leveraged the strengths of each. The underlying idea was to treat each individual model as a "voter" or "coach" in a national poll, with some voices weighted more heavily depending on how well they predicted actual top teams.

Next, to enable comparison across models that use different scoring scales, I standardized each model's output using z scores (mean of 0 and standard deviation of 1). This step allowed me to combine them without any single model's raw scale dominating the others.

To objectively determine how much weight each model should carry in the final ranking, I used a logistic regression framework. I identified a binary outcome variable labeled `Top_25`, indicating whether each team finished in the actual final top 25 rankings. This gave me a real world benchmark to anchor my modeling.

18.1 Logisitc Regression Model

The logistic regression model used the standardized scores from each ranking method as predictor variables and the `Top_25` indicator as the outcome. By examining the absolute value of the resulting coefficients (excluding the intercept), I could extract how important each model was in determining whether a team was truly among the best.

These coefficients were normalized to sum to one, creating a set of weights (`alphas`) that reflected each model's predictive utility. For instance, if the Elo model had a high coefficient, it received a larger share of the final ranking calculation.

The final ranking score for each team was computed as a weighted average of the standardized scores from each model, multiplied by their corresponding `alpha` weight. This aggregated score represents the most holistic view of a team's performance across all the lenses I explored.

Finally, the teams were sorted by this composite score to generate a unified, ranked list that reflects contributions from all component models, optimally balanced based on how well each model reflected true performance. This methodology ensures that the final ranking is not reliant on any one model but instead represents a consensus informed by performance, margin, connectivity, and prior expectations.

This final model did a much better job at ranking teams in alignment with how well they actually performed over the course of the season. Notably, it also corrected for some of the earlier limitations seen in individual models. For example, Occidental, who was ranked in the early 100s in the basic Markov Chain model, rose to the 50s in the final model, reflecting a more accurate placement based on their overall season. This demonstrated that combining the strengths of multiple models into a single framework provided a clearer and more balanced view of team quality.

18.2 Ridge Regression Ranking Model

While the composite model based on logistic regression coefficients gave a robust way to merge multiple ranking systems, I wanted to know whether or not an even more effective method could be discovered through enhancing the way that the weights for every model were determined. Still on the idea that each model could be viewed as a "coach" or voter in an opinion poll, I experimented with Ridge Regression, a regularized form of linear modeling that discourages large coefficient values to avoid overfitting while still allowing all predictors to be utilized.

Ridge regression allowed me to develop a model in which all ranking sets could contribute somewhat to the eventual output, yet those that failed to be quality predictors of actual top 25 teams would also have small weights, maybe, towards zero. This was the key to allowing multiple perspectives yet preventing noisy and weak predictors to dominate the end ranking.

I began by building this model using the feature matrix constructed from each model I had originally built: Elo, Bayesian, PageRank, Filtered Transition, Win Margin, Low Favor Ties, and Strength of Schedule. The binary response variable was employed to denote whether a team actually finished in the top 25.

I then fit a Ridge logistic regression model with cross validation to estimate the optimal penalty term, , that controls how strong the regularization should be. I then extracted the model's coefficients and normalized their absolute values to create a second set of weights, now regularized by the Ridge machinery. The normalized coefficients acted similar to "alpha weights," but similar to the previous model with additional shrinkage that reduced weight noisy predictors.

The final normalized Ridge weights assigned to each model were:

- **PageRank:** 0.1551
- **Elo Rating:** 0.1326
- **Filtered Transition:** 0.0023
- **Win Margin:** 0.0723
- **Low Favor Ties:** 0.0632
- **Strength of Schedule:** 0.1577
- **Bayesian:** 0.0896
- **Final Logistic Composite Score:** 0.1670

These weights show that the Ridge model significantly downweighted the Filtered Transition model, assigning it very little influence in the final ranking, while placing greater emphasis on metrics such as Strength of Schedule, PageRank, and the combined logistic model. The Elo, Bayesian, and Win Margin models also retained meaningful influence.

Using these weights, I calculated a new composite score for each team, producing a final set of Ridge weighted rankings. This method provided another rigorous, regularized approach to integrating multiple perspectives on team strength, and gave slightly more reliable rankings compared to the earlier models.

By allowing the model to automatically downweight less useful predictors while retaining all sources of information, the Ridge regression model provided an even more disciplined and potentially accurate way of consolidating the perspectives of all the models I developed throughout this project.

18.3 Evaluating Model Accuracy: Logistic vs. Ridge Regression

Both the logistic regression model and the Ridge regression model proved to be highly effective in identifying the top 25 teams based on their season performance. Each model correctly predicted 18 of the 25 teams that ended up in the actual top 25, a strong outcome given the variability in match results and team scheduling.

To better assess how the models performed, I ran a series of evaluation metrics. For the Ridge regression model, I used its fitted probability scores and converted them to binary predictions using a decision threshold of 0.35. The threshold of 0.35 was chosen instead of the conventional 0.5 to improve sensitivity in detecting top performing teams. Because top 25 teams are relatively rare in the dataset compared to the total number of teams, using a 0.5 cutoff risked being too conservative and missing teams that were likely top 25 candidates but had slightly lower predicted probabilities. Lowering the threshold to 0.35 increased the model's recall, its ability to correctly identify true positives, without severely compromising precision. This allowed for a more inclusive prediction window that better aligned with the goal of identifying high performing teams based on available data. A team was classified as a top 25 team if its predicted probability exceeded this threshold. I then compared these predictions against the actual list of top 25 teams.

The confusion matrix from this classification showed the following results:

- True Positives (correctly predicted top 25 teams): 20
- False Positives (teams incorrectly included in the top 25): 9
- False Negatives (actual top 25 teams missed): 5
- True Negatives (teams correctly excluded from the top 25): 382

Using this, I calculated several standard metrics for evaluating binary classification performance:

- **Precision:** 0.69
- **Recall:** 0.80
- **Accuracy:** 0.966

These metrics confirm that the Ridge model was both accurate and precise, capturing a strong proportion of true top teams while avoiding many false positives.

After assessing overall predictive accuracy, I also compared the final rankings from both the logistic and Ridge models. I calculated the difference in rankings assigned to each team by the two models to see which teams moved the most. The Ridge model not only showed strong predictive metrics but also exhibited better placement of good teams higher in the rankings and kept weaker teams from being overvalued compared to the logistic regression model. For example, teams that were closer to the true top 25 were generally placed more accurately under Ridge, while the logistic regression model sometimes ranked weaker teams higher than they should have been.

Overall, the Ridge regression model provided a slightly more refined and consistent ranking system by penalizing unreliable predictors and emphasizing models that demonstrated strong predictive ability in cross validation.

18.4 Validating Model Weights on Simulated Data

To further evaluate whether the Ridge regression model accurately captures the best teams, I applied it to the simulated dataset where the true top teams were explicitly defined. This was the same dataset used earlier, where all SCIAC teams were assumed to be the strongest teams in the country, with every SCIAC team assigned a scoring lambda of 2.5 and all other teams assigned a lambda of 1.0. This setup was meant to mimic a world in which SCIAC teams consistently outperform their opponents, allowing for a controlled environment to test whether the Ridge regression weights still prioritize the correct teams.

After generating match outcomes using this simulated data, I ran each individual model, Elo, PageRank, filtered transition, win margin, low favor ties, strength of schedule, and Bayesian, on the new dataset. I then applied the exact same Ridge regression weights that were previously learned from the real world data. The idea was to check whether the final composite rankings, constructed from these fixed weights, successfully elevated the SCIAC teams as the top performers, given their built in statistical advantage in the simulated matches.

This experiment was a way of testing the robustness and generalizability of the Ridge derived model. If the weights were effective, SCIAC teams should appear near the top of the final rankings, confirming that the Ridge model is not just overfitting to the real world dataset but can also generalize to data where the true hierarchy of teams is known. Indeed, when I applied the weights to the simulated results, SCIAC teams all appeared prominently in the top 15 of the rankings, validating that the weighting scheme accurately reflects team strength across different data conditions.

18.5 Simulated Dataset with Known Team Strengths

To further validate the reliability of the ridge regression model, I created a simulated dataset in which I could directly control and know the true rankings of every team. Specifically, I assigned a unique Poisson rate parameter (λ) to each of the 427 teams, where each λ was randomly drawn from a uniform distribution between 0.5 and 4.5. This parameter governed how frequently each team would score in a given match, thereby representing its underlying strength.

Simulation Setup

First, I filtered the dataset to include only teams that had played at least three games, ensuring that each team had a sufficient number of matches to reasonably estimate their strength. I then identified all unique teams that remained after this filtering step.

Next, I assigned each team a random λ value using a uniform distribution:

$$\lambda_i \sim \text{Uniform}(0.5, 4.5)$$

This distribution was chosen intentionally. By capping the upper end at 4.5, the simulation stayed within a realistic scoring range for soccer, where extremely high scores are rare. This ensured that goal margins and scoring frequencies remained interpretable and appropriate for the models being tested, many of which explicitly incorporate scoring margin or expected goals into their calculations.

Using these assigned λ values, I simulated new match results by drawing scores from a Poisson distribution:

$$\text{Goals}_{\text{home}} \sim \text{Poisson}(\lambda_{\text{home}}), \quad \text{Goals}_{\text{away}} \sim \text{Poisson}(\lambda_{\text{away}})$$

These simulated scores were then used to reconstruct a new dataset with columns for home and away team names, goals scored by each team, and goal margins.

Ground Truth and Realism

Once this dataset was created, I preserved the assigned λ values in a new dataframe, `lambda_truth`, which provided the ground truth ranking of teams from strongest (highest λ) to weakest (lowest λ). This gave me a definitive standard against which to measure how accurately each model, and especially the ridge weighted ensemble, was able to recover the true team ordering.

It is worth noting that assigning λ values across 427 teams inevitably led to many teams having very similar scoring rates. In practice, this mirrors the reality of Division III college soccer, where the skill gap between teams is often narrow. As a result, the problem of accurately ranking teams based on match outcomes becomes especially challenging, the stochastic nature of the Poisson draws, compounded by minimal differences in underlying strength, makes it difficult for any model to perfectly recover the true ranking.

Nonetheless, this framework provides a powerful controlled setting: I know the "correct" team order and can objectively evaluate how closely any ranking method approximates it. In the next section, I assess how well the ridge regression model performed on this simulated dataset using the same ridge weights derived from the real world data, thereby testing the generalizability of that learned weighting scheme.

18.6 Incorporating Multidimensional Team Strength: Offensive and Defensive Simulation

One of the challenges that became increasingly apparent as I worked on this simulating data to evaluate my model was the inherent complexity of soccer as a sport. Unlike the previous simulated dataset wI used where there was a single λ assigned to each team, the reality is that a team's performance cannot be boiled down to a one dimensional score. Soccer is deeply multidimensional.

Each team fields 11 players, all of whom have different roles, responsibilities, and skill sets. Even at the individual level, a player's capabilities can be broken down into dozens of attributes, such as speed, passing, finishing, positioning, and defensive awareness, each of which contributes in different ways to match outcomes. Thinking in terms of video games like FIFA, each player is often evaluated using 30 or more discrete ratings, which together determine a composite value. Consequently, reducing entire teams to a single rating misses much of this nuance.

To better capture this complexity, I extended the earlier simulation framework by incorporating two dimensions of team strength: offense and defense. Rather than assigning a single Poisson rate parameter λ to each team, I assigned each team an *offensive* and a *defensive* λ value. These were randomly drawn from the same uniform distribution as before:

$$\lambda_{\text{offense}} \sim \text{Uniform}(0.5, 4.5), \quad \lambda_{\text{defense}} \sim \text{Uniform}(0.5, 4.5)$$

This setup maintains the realism of goal scoring and game dynamics, while beginning to reflect the multi dimensional nature of real soccer. The final simulated goal expectation for each team in a given match was computed as:

$$\lambda_{\text{home}} = \frac{\lambda_{\text{offense, home}}}{\lambda_{\text{defense, away}}}, \quad \lambda_{\text{away}} = \frac{\lambda_{\text{offense, away}}}{\lambda_{\text{defense, home}}}$$

This formulation ensures that both a team's ability to attack and the opponent's ability to defend influence scoring outcomes, which more closely mirrors real world match dynamics. For example, a strong offensive team may still struggle to score if the opposing defense is particularly effective, and vice versa.

Using these adjusted λ values, I once again simulated a full season's worth of matches. Each game was generated using the Poisson distribution, as in prior experiments, but this time driven by the adjusted offense/defense interactions. I then ran all seven individual ranking models on this simulated dataset and applied the same ridge regression weights that had been trained using the real world data. This allowed me to assess how well the original model, with fixed weights, was able to recover the known true team rankings from a more realistic, two dimensional simulation.

The goal of this extension was to evaluate whether the original ridge based ensemble generalizes well in more complex, structured environments. While soccer involves far more than just offensive and defensive ability, adding even this simple second dimension represented a significant step toward capturing the richness of the game. The more dimensions incorporated into a simulation, the closer it moves toward emulating the intricacies of real world team interactions.

18.6.1 Verifying Simulation Realism

Before applying my ensemble model to the newly simulated data with offensive and defensive values, it was crucial to verify that the simulation accurately replicated the statistical patterns of real world match data. After all, if the simulated outcomes bore no resemblance to real game results, any conclusions drawn from evaluating model performance on that dataset would be suspect. In other words, I needed to ensure that the simulation not only followed the right logic but also produced realistic soccer outcomes.

To accomplish this, I created a set of diagnostic visualizations comparing two key metrics from the real and simulated datasets: the distribution of absolute score differentials and the distribution of team win percentages. The first histogram compares how often teams win by various margins, a fundamental reflection of how competitive or lopsided matches are across a season. The second captures how win percentages are distributed across all teams, providing insight into parity or dominance within the simulated league structure.

In both plots, I overlaid the real NCAA data with the simulated dataset using stacked histograms. As the graphs illustrate, there was a strong alignment between the two distributions. In the score differential plot, both datasets are heavily concentrated around narrow margins, typically one or two goal results, with very few games decided by larger differences. This is consistent with real soccer outcomes, where close games are the norm. Likewise, the win percentage histogram shows that the simulated data mirrors the skew and spread seen in real team records, including the presence of highly successful and struggling teams.

These comparisons gave me confidence that the simulation was performing as intended, producing realistic scores, outcomes, and team records that reflected the statistical dynamics of a real NCAA soccer season. Only after confirming this fidelity did I move forward and apply the ridge regression model, trained on real world data, to this simulated environment. With this foundation in place, I could evaluate not only how well the model performed, but also whether it generalized appropriately to a setting where the ground truth team strengths were explicitly known.

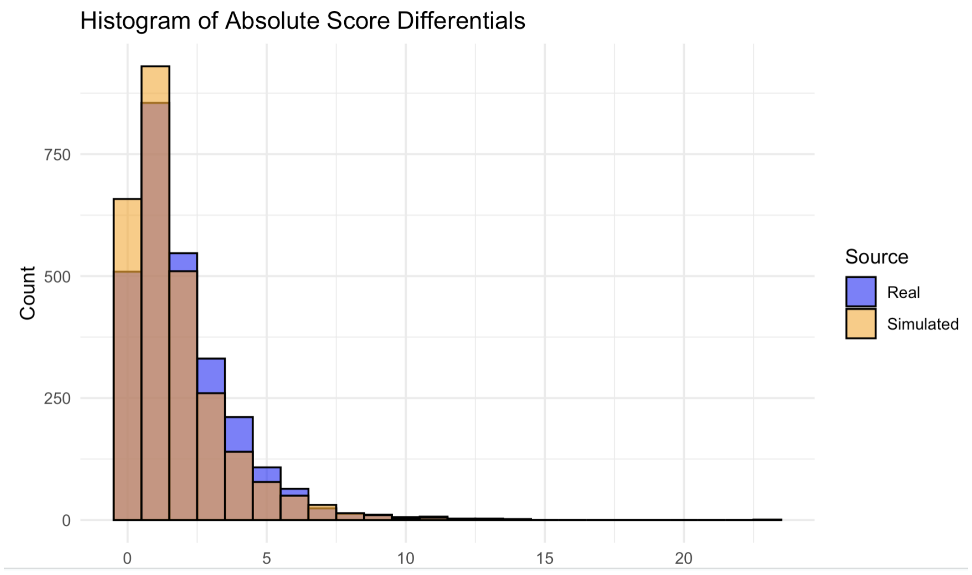


Figure 3: Histogram of Absolute Score Differentials Real vs. Simulated

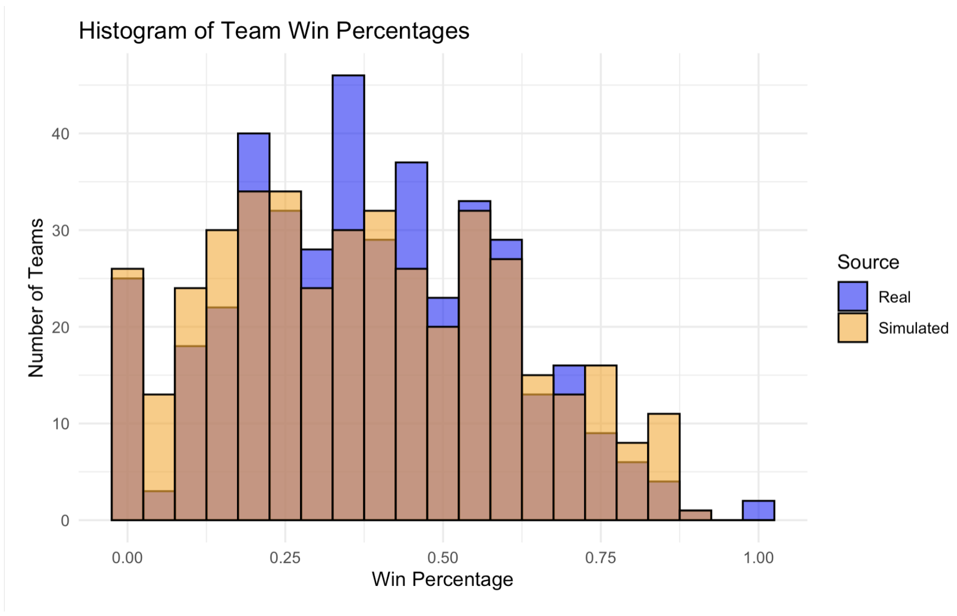
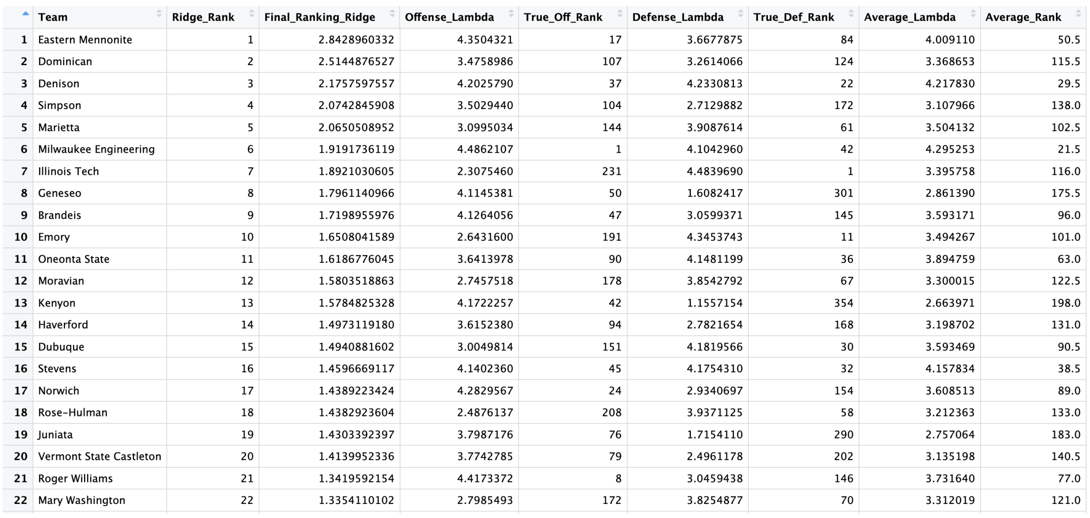


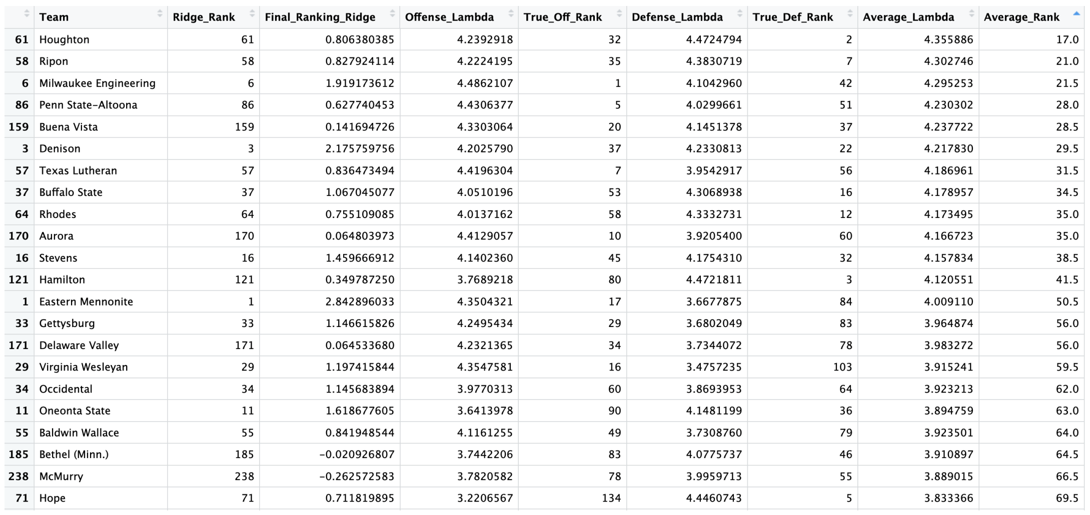
Figure 4: Histogram of Team Win Percentages Real vs. Simulated

18.7 Results



	Team	Ridge_Rank	Final_Ranking_Ridge	Offense_Lambda	True_Off_Rank	Defense_Lambda	True_Def_Rank	Average_Lambda	Average_Rank
1	Eastern Mennonite	1	2.8428960332	4.3504321	17	3.6677875	84	4.009110	50.5
2	Dominican	2	2.5144876527	3.4758986	107	3.2614066	124	3.368653	115.5
3	Denison	3	2.1757597557	4.2025790	37	4.2330813	22	4.217830	29.5
4	Simpson	4	2.0742845908	3.5029440	104	2.7129882	172	3.107966	138.0
5	Marietta	5	2.0650508952	3.0995034	144	3.9087614	61	3.504132	102.5
6	Milwaukee Engineering	6	1.9191736119	4.4862107	1	4.1042960	42	4.295253	21.5
7	Illinois Tech	7	1.8921030605	2.3075460	231	4.4839690	1	3.395758	116.0
8	Geneseo	8	1.7961140966	4.1145381	50	1.6082417	301	2.861390	175.5
9	Brandeis	9	1.7198955976	4.1264056	47	3.0599371	145	3.593171	96.0
10	Emory	10	1.6508041589	2.6431600	191	4.3453743	11	3.494267	101.0
11	Oneonta State	11	1.6186776045	3.6413978	90	4.1481199	36	3.894759	63.0
12	Moravian	12	1.5803518863	2.7457518	178	3.8542792	67	3.300015	122.5
13	Kenyon	13	1.5784825328	4.1722257	42	1.1557154	354	2.663971	198.0
14	Haverford	14	1.4973119180	3.6152380	94	2.7821654	168	3.198702	131.0
15	Dubuque	15	1.4940881602	3.0049814	151	4.1819566	30	3.593469	90.5
16	Stevens	16	1.4596669117	4.1402360	45	4.1754310	32	4.157834	38.5
17	Norwich	17	1.4389223424	4.2829567	24	2.9340697	154	3.608513	89.0
18	Rose-Hulman	18	1.4382923604	2.4876137	208	3.9371125	58	3.212363	133.0
19	Juniata	19	1.4303392397	3.7987176	76	1.7154110	290	2.757064	183.0
20	Vermont State Castleton	20	1.4139952336	3.7742785	79	2.4961178	202	3.135198	140.5
21	Roger Williams	21	1.3419592154	4.4173372	8	3.0459438	146	3.731640	77.0
22	Mary Washington	22	1.3354110102	2.7985493	172	3.8254877	70	3.312019	121.0

Figure 5: This table shows the top 22 teams as ranked by the final ridge regression model applied to simulated match data, where each team was randomly assigned an offensive and defensive lambda value between 0.5 and 4.5. The columns display each team's ridge rank and final score, along with their true offensive and defensive lambda values and corresponding ranks. The average of these lambdas provides an estimated true team quality. This visualization allows for direct comparison between how the ridge model ranked teams and their true underlying simulated strength, helping evaluate how well the model generalized under a realistic two dimensional soccer simulation.



	Team	Ridge_Rank	Final_Ranking_Ridge	Offense_Lambda	True_Off_Rank	Defense_Lambda	True_Def_Rank	Average_Lambda	Average_Rank
61	Houghton	61	0.806380385	4.2392918	32	4.4724794	2	4.355886	17.0
58	Ripon	58	0.827924114	4.2224195	35	4.3830719	7	4.302746	21.0
6	Milwaukee Engineering	6	1.919173612	4.4862107	1	4.1042960	42	4.295253	21.5
86	Penn State–Altoona	86	0.627740453	4.4306377	5	4.0299661	51	4.230302	28.0
159	Buena Vista	159	0.141694726	4.3303064	20	4.1451378	37	4.237722	28.5
3	Denison	3	2.175759756	4.2025790	37	4.2330813	22	4.217830	29.5
57	Texas Lutheran	57	0.836473494	4.4196304	7	3.9542917	56	4.186961	31.5
37	Buffalo State	37	1.067045077	4.0510196	53	4.3068938	16	4.178957	34.5
64	Rhodes	64	0.755109085	4.0137162	58	4.3332731	12	4.173495	35.0
170	Aurora	170	0.064803973	4.4129057	10	3.9205400	60	4.166723	35.0
16	Stevens	16	1.459666912	4.1402360	45	4.1754310	32	4.157834	38.5
121	Hamilton	121	0.349787250	3.7689218	80	4.4721811	3	4.120551	41.5
1	Eastern Mennonite	1	2.842896033	4.3504321	17	3.6677875	84	4.009110	50.5
33	Gettysburg	33	1.146615826	4.2495434	29	3.6802049	83	3.964874	56.0
171	Delaware Valley	171	0.064533680	4.2321365	34	3.7344072	78	3.983272	56.0
29	Virginia Wesleyan	29	1.197415844	4.3547581	16	3.4757235	103	3.915241	59.5
34	Occidental	34	1.145683894	3.9770313	60	3.8693953	64	3.923213	62.0
11	Oneonta State	11	1.618677605	3.6413978	90	4.1481199	36	3.894759	63.0
55	Baldwin Wallace	55	0.841948544	4.1161255	49	3.7308760	79	3.923501	64.0
185	Bethel (Minn.)	185	−0.020926807	3.7442206	83	4.0775737	46	3.910897	64.5
238	McMurry	238	−0.262572583	3.7820582	78	3.9959713	55	3.889015	66.5
71	Hope	71	0.711819895	3.2206567	134	4.4460743	5	3.833366	69.5

Figure 6: This table displays the top teams based on their true average lambda values, which combine independently assigned offensive and defensive lambda ratings for each team. These represent each team's actual simulated strength in the dataset. The table allows comparison with the Ridge regression ranks to evaluate how closely the model's rankings align with the known ground truth. While many top true teams (e.g., Eastern Mennonite, Denison, Stevens) are ranked highly by the model, others such as Aurora and Penn State–Altoona are ranked lower than expected. This highlights both the effectiveness and limits of the ridge model under match randomness, realistic soccer score simulation, and minor differences between team strengths.

18.8 Model Conclusions

In gauging the performance of my model on real NCAA Division III soccer data, it initially appeared quite strong. The model got 18 out of the actual top 25 teams correct, suggesting that it had learned to replicate the sorts of patterns and signals that the NCAA uses in determining national rankings. That is, it was very good at recognizing what the system sees as strong teams, teams with good records, difficult opponents, and results that fit the narrative script. But when I used this same model to evaluate test data simulated on objective offensive and defensive strength measures for each team and match results from a Poisson distribution, the story was different. This time, the model correctly identified only 3 of the actual top 25 teams, even though the schedule format was the same.

This difference uncovers something more profound: the model wasn't really learning to recognize actual team strength, it was learning to simulate the NCAA's sense of strength. That sense, as it happens, is strongly influenced by outside and structural considerations, rather than by on field results. Particularly, the SCIAC conference is structurally disadvantaged in this regard. Geographic remoteness limits interregional matchups, thereby lowering SCIAC teams' Opponent Win Percentage (OWP) and Opponent's Opponent Win Percentage (OOWP), both of which form the foundation of strength of schedule measures used by the NCAA. Even dominating in the SCIAC is not enough to raise a team's perceived level of strength because that dominance is in an undervalued and isolated system.

The model's inability to find true strength in the simulated world is reflective of the NCAA's own blind spots, both are operating within systems that take into account perceived but not actual ability. And because every ranking system ends up relying on assumptions that it builds about what "strength" is, any system that doesn't consider structural isolation will continue to underrate teams like those in the SCIAC. What looks at first glance to be a malfunction of the model is in reality a glimpse of a broader system bias, a bias that obtains regardless of whether a human panel or a machine learning program makes the judgment.

19 Conclusion

For the 2024–2025 season, the NCAA adopted the NCAA Power Index (NPI) as a new Division III soccer rankings system. The formula based approach was designed to bring greater consistency and objectivity to the selection and seeding process for the tournament based on winning percentage, strength of schedule, quality win bonuses, and home away adjustments. The NPI's flexibility, especially through its calibrated "dials," just recognizes the NCAA's acknowledgment that a single figure cannot possibly describe team performance for a Division III soccer universe so large and geographically expansive as this.

While I am not suggesting the NPI should be replaced with what I've crafted, the merit of the latter lies in being able to lend a hand with diagnosing imperfections in today's system. My examination finds that models trained on actual outcome results do an excellent job of replicating NCAA rankings, but in doing so, the success also reveals an issue: they are identifying perceived strength, rather than actual team quality. Tested against simulated data where team strength was defined objectively via offensive and defensive measures, my model failed to recognize the best teams. The same schedule produced wholly different outcomes because the existing system, and my model, trained on it, undervalues structurally disadvantaged teams.

This finding has direct implications for teams in geographically isolated conferences like the SCIAC. Despite perennial postseason achievements, SCIAC teams are still underrepresented nationally and receive few, if any, at large bids. My research verifies the argument that their isolation is not just logistical; it's numerical. Strength of schedule metrics like OWP and OOWP, which drive much of the system nowadays, simply can't factor in the quality of a team if they barely ever get the opportunity to play nationally recognized opponents.

Throughout this project, I was able to speak with the SCIAC representative on the NCAA national selection committee. I was explaining some of the early results that I'd noted, and she explained that adopting the NPI was in one way a response to precisely those same challenges that I'd begun to study, an effort to move the system in the right direction. She said the NPI is in no sense a perfect mechanism but that it represented an actual movement toward fairness of process. It was comforting to discover that the problems I encountered throughout this project were not only identified but were already beginning to shape change for the future of how division 3 soccer is ranked.

Lastly, I hope that this thesis can be something more than a technical examination of ranking algorithms. My goal is that it provides evidence based support for why certain teams, particularly those that face structural disadvantages, deserve a more accurate and fair method of evaluation. By simulating ground truth and comparing it to both human and model generated rankings, I've shown where the current system falls short and where new methods could begin to close the gap. I hope that this work not only gives aid to current improvement of the NPI, but also moves the conversation further, toward a future where all teams, regardless of where they come from, have an equal opportunity at recognition and opportunity.

References

[1] Léon Bottou, *Stochastic gradient descent tricks*, Neural Networks: Tricks of the Trade (Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, eds.), Springer, Berlin, Heidelberg, 2 ed., 2021, pp. 421–436.

[2] Gabriel Chandler, *Evaluating learners and ensemble methods*, Lecture 18, Computational Statistics, Pomona College, November 2022.

[3] Nikhil Ghosh, Spencer Frei, Wooseok Ha, and Bin Yu, *The effect of sgd batch size on autoencoder learning: Sparsity, sharpness, and feature learning*, arXiv preprint arXiv:2308.03215 (2023).

[4] Geoffrey E. Hinton and Richard R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science **313** (2006), no. 5786, 504–507.

[5] Nikhil Ketkar, *Stochastic gradient descent*, pp. 113–132, Apress, 2017.

[6] Ravi P. Kumar, Alex K.L. Goh, and Ashutosh K. Singh, *Application of markov chain in the pagerank algorithm*, Pertanika Journal of Science & Technology **21** (2013), no. 1, 541–554.

[7] Michael A. Lapré and Elizabeth M. Palazzolo, *Quantifying the impact of imbalanced groups in fifa women's world cup tournaments 1991–2019*, Journal of Quantitative Analysis in Sports **18** (2022), no. 3.

[8] Jan Lasek and Marek Gagolewski, *Interpretable sports team rating models based on the gradient descent algorithm*, International Journal of Forecasting **37** (2021), 1061–1071.

[9] Nicole R. Matthews, Andrew McClain, Chase M.L. Smith, and Adam G. Tennant, *Application of pagerank algorithm to division i ncaa men's basketball as bracket formation and outcome predictive utility*, Journal of Sports Analytics **7** (2021), 1–9.

[10] NCAA, *Ratings percentage index (rpi): Basics and formula*, https://www.ncaa.com/news/basketball-men/article/2019-03-12/mens-basketball-net-rankings-explained, 2019, Accessed April 2025.

[11] Sheldon M. Ross, *Simulation*, 4th ed., Academic Press, Burlington, MA, 2006.

[12] A. Shapiro and Y. Wardi, *Convergence analysis of gradient descent stochastic algorithms*, Journal of Optimization Theory and Applications **91** (1996), 439–454.

[13] Joel Sokol and James Smith, *Ranking ncaa basketball teams using markov chains*, School of Industrial and Systems Engineering, Georgia Institute of Technology (2007).