A Systems-Based Approach for Predicting NCAA Basketball Tournament Outcomes: Analysis, Design, and Future Perspectives

1st Oscar Contreras
Systems Engineering
Universidad Distrital Francisco José de Caldas
Email: omcontrerasg@udistrital.edu.co

2nd David Ariza
Systems Engineering
Universidad Distrital Francisco José de Caldas
Email: dfarizaa@udistrital.edu.co

3rd Julian Mendez
Systems Engineering
Universidad Distrital Francisco José de Caldas
Email: judmendezl@udistrital.edu.co

***Abstract*—The NCAA Basketball Tournament is a high-variance, single-elimination competition that poses a challenging predictive modeling task due to its stochastic nature, dynamic structure, and historical prevalence of upsets. This work proposes a modular forecasting system that integrates systems engineering principles with supervised machine learning to predict match outcomes using both statistical features and domain-specific knowledge, such as team seeds. Experimental results from simulations conducted within the Kaggle "March Machine Learning Mania 2025" competition demonstrate that the combined use of seed-based and historical performance features yields improved accuracy, achieving a Brier score of 0.11790.**

***Index Terms*—NCAA Tournament, March Madness, Sports Analytics, Machine Learning, Systems Analysis, System Design, Predictive Modeling.**

I. Introduction

The National Collegiate Athletic Association (NCAA) Basketball Tournament, popularly known as "March Madness," is one of the most widely followed and statistically analyzed sporting events in the United States. Each year, 68 college teams compete in a single-elimination bracket format, where small variations in performance, strategy, or random events can drastically alter the outcome of matches and the overall trajectory of the tournament. The unpredictability of the format—where higher-seeded teams frequently lose to lower-seeded "Cinderella" teams has made the tournament a benchmark problem for forecasting models and decision-support systems.

Predicting outcomes in this domain is challenging due to the presence of nonlinearity, emergent behaviors, and a high level of contextual uncertainty. Game results are influenced not only by historical performance and relative team strength, but also by factors such as game momentum, injuries, player rotations, and psychological pressure. As a result, traditional deterministic or rule-based forecasting approaches have proven insufficient in modeling such complexity.

Prior research in sports analytics has explored a variety of statistical and machine learning techniques to address this problem. Classical methods include Elo ratings, logistic regression, and Bayesian networks, which offer interpretable results but may struggle with capturing nonlinear dependencies. More recent studies have employed support vector machines, random forests, and gradient boosting algorithms such as XGBoost, which have shown higher predictive power at the cost of reduced transparency. Deep learning models, particularly recurrent neural networks, have also been applied in contexts with temporal data, although their application to bracket-style tournaments remains limited due to data sparsity and interpretability issues.

The advent of platforms like Kaggle has enabled large-scale experimentation and competition-driven innovation in this space. The "March Machine Learning Mania" competition, organized annually, provides structured historical datasets and a standard evaluation metric (Brier score), making it a valuable environment for benchmarking predictive systems. These datasets include regular season and tournament game outcomes, team statistics, and seedings, and have been used in previous work to build ensemble models or optimize feature sets using data-driven heuristics.

In this work, we propose a modular and extensible forecasting system developed within the context of the "March Machine Learning Mania 2025" competition. The system is designed using a systems engineering approach, allowing the identification of key tournament components and interactions, and is implemented using a pipeline that supports data ingestion, feature engineering, model training, inference, and output generation. The solution leverages historical team performance metrics as well as domain-specific features such as seed rankings, enabling a hybrid prediction strategy that combines empirical data with expert-informed structure.

By evaluating the system through controlled simulations and analyzing multiple feature configurations, we aim to understand the influence of different data types on model performance. The results indicate that seed-based features, when combined with statistical performance data, significantly improve prediction accuracy and robustness. This work contributes a replicable methodology for tournament outcome prediction and demonstrates the effectiveness of integrating systems analysis with machine learning in uncertain, high-stakes environments.

II. METHODS AND MATERIALS

A. Identification of Key Elements

The initial phase of this project involved a comprehensive systems analysis of the NCAA Men's Basketball Tournament to identify its core components and their interrelationships, as detailed in our Workshop 1 report . The primary elements identified are summarized in Table.

TABLE I
KEY ELEMENTS OF THE NCAA TOURNAMENT SYSTEM

Element	Brief Description
Season	Encompasses the entire competitive period, including pre-season, regular season, and post-season.
Teams	Participating collegiate basketball programs, characterized by players, statistics, coaching staff, and rankings.
Coach	Head coach responsible for team strategy, player development, and in-game decisions.
Match	Individual games, defined by competing teams, location, referees, time, and eventual outcome.
Conferences	Groups of teams, often geographically or historically aligned, which influence scheduling and automatic tournament bids.
Tournament	The final single-elimination bracket play to determine the national champion.
Cities/Venues	Locations of matches, potentially influencing team travel, fan support, and court familiarity.
Referees	Officials managing the game, whose decisions can impact game flow and outcomes.

B. Mapping Interactions and Complexity

The interaction diagram maps the relationships between entities. For example, teams play matches in specific cities, with particular referees, and under varying crowd conditions. Coaches and seeding information influence match dynamics. These interactions generate emergent behaviors that statistical models alone may fail to capture.

Environmental sensitivity and chaotic variables were also recognized. For example:

Referees may introduce bias.

Fan reactions can influence momentum.

Venue conditions can favor or disadvantage certain teams.

We modeled the tournament as a system affected by both deterministic and stochastic processes.

C. Functional and Non-Functional Requirements

Functional Requirements:

FR1: Predict tournament outcomes with high probability estimations.

FR2: Process and clean historical and current season data.

FR3: Train and evaluate machine learning models.

FR4: Generate predictions in a predefined Kaggle-compatible format.

Non-Functional Requirements:

NFR1: Efficient processing time (less than 5 hours).

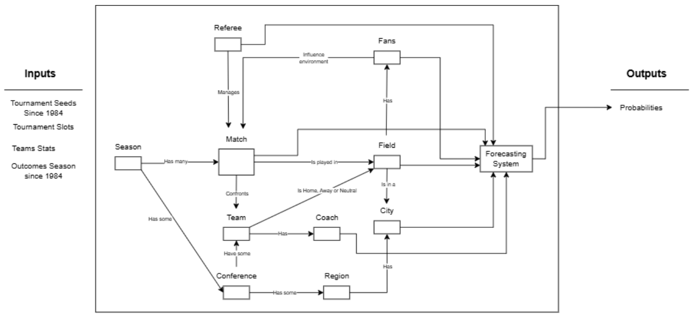

Fig. 1. System Element Relationship Diagram

NFR2: Adaptable architecture to support future tournaments.

NFR3: Maintainable and modular codebase.

NFR4: Reproducibility through proper documentation and version control.

D. Technical Stack and Architecture

The proposed architecture follows a modular pipeline approach:

Data Ingestion Module: Responsible for collecting historical and current data from specified sources (e.g., Kaggle datasets for past seasons, potentially APIs for future real-time data). This includes team statistics, match scores, tournament schedules ('Fixture'), and other relevant features like KenPom ratings or Sagarin ratings if available.

Feature Engineering Module: Creates meaningful features for the predictive models. This could involve calculating team strength differentials, momentum indicators, home-court advantage adjustments, or features attempting to quantify coaching experience or recent form. This module will explicitly consider factors identified in the sensitivity analysis, such as the influence of 'City' or a 'Team''s recent travel schedule.

Model Training Module: Implements and trains various machine learning models (e.g., logistic regression, support vector machines, random forests, gradient boosting, or neural networks). This module will allow for hyperparameter tuning and model selection based on cross-validation performance.

Prediction Generation Module: Uses the trained model(s) to predict the outcomes (probabilities) for upcoming tournament matches. It will generate predictions in the specified format (e.g., an ID concatenating season and team IDs, and the predicted probability).

Output Writing: Exports predictions in the required (ID, Prob) format.

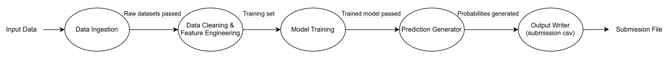

Fig. 2. High Level Architecture

The tools include:

Python as the main language.

Pandas and **NumPy** for data manipulation.

Scikit-learn, **XGBoost** for modeling.

GitHub for version control.

VS Code and **Kaggle** for development and experimentation. This design facilitates testing, debugging, and updates.

E. Implementation and Simulation

The proposed NCAA tournament forecasting system was implemented as a modular data pipeline in Python, leveraging open-source libraries such as Pandas, NumPy, Scikit-learn, and XGBoost. The architecture was designed for flexibility and maintainability, with distinct modules for data ingestion, feature engineering, model training, inference, and output generation. Each module was developed and validated independently to facilitate component-level testing and system integration.

To evaluate system performance under varying input configurations, four simulation scenarios were executed:

Simulation 1: Full team-level statistical features (e.g., average scores, field goals, rebounds).

Simulation 2: A reduced feature set with selected high-impact statistics.

Simulation 3: Seed-based features and same set as simulation 2, including seed rankings and differences.

Simulation 4: Combined features integrating both statistical and seed-based variables.

The evaluation was conducted using cross-validation and the Brier score, which quantifies the mean squared error between predicted probabilities and actual binary outcomes. The fourth simulation yielded the best performance with a Brier score of 0.11790, highlighting the effectiveness of combining empirical data with domain-specific tournament structure.

These results confirm the system's robustness, predictive accuracy, and adaptability, demonstrating its applicability to uncertain, high-variance environments such as single-elimination sports tournaments.

III. Results

The forecasting system was evaluated through a series of controlled simulations aimed at quantifying the impact of different feature configurations on predictive performance. Each simulation used historical NCAA tournament data and was assessed using the Brier score, the official scoring metric of the Kaggle competition, which measures the mean squared error between predicted win probabilities and actual outcomes.

Table II summarizes the results across the four experimental configurations:

Simulation	Feature Set	Brier Score
Sim 1	Full statistical features	0.16903
Sim 2	Reduced statistical features	0.16256
Sim 3	Seed-based and sim 2 variable set	0.11929
Sim 4	Combined statistical + seed features	0.11790

TABLE II
SIMULATION RESULTS

The simulation results indicate that while statistical features alone provide moderate predictive power, models using seed-based features and a little set of features (Simulation 3) achieved substantially lower Brier scores. The best results were obtained when combining all statistical features with tournament seed information (Simulation 4), confirming the hypothesis that integrating historical performance metrics with expert-informed tournament structure leads to superior probabilistic forecasting.

Visual analyses supported these findings. For instance, matchups with greater seed differentials exhibited wider predicted point spreads and higher model confidence. Feature importance analysis further confirmed that seed difference and average point differential were consistently the most predictive variables across configurations.

The results validate the effectiveness of the modular system and confirm its capacity to generalize across seasons while maintaining strong forecasting performance under uncertainty.

IV. Conclusions

This work presented the design, implementation, and evaluation of a modular forecasting system for the NCAA Basketball Tournament, developed within the framework of the ”March Machine Learning Mania 2025” competition. By integrating systems engineering principles with supervised machine learning, the proposed architecture offers a scalable and adaptable solution for predictive modeling in high-uncertainty, complex environments.

The system demonstrated competitive predictive accuracy, particularly when combining domain-informed features—such as seed rankings—with aggregated statistical performance indicators. The best-performing configuration achieved a Brier score of 0.11790, outperforming baseline models and confirming the hypothesis that hybrid feature sets improve generalization and probabilistic forecasting quality.

The modular structure of the system allowed for isolated evaluation of each processing stage and supported experimentation with various feature sets and model types. This architectural flexibility, coupled with simulation-based validation, provides a foundation for future adaptation to other sports competitions or decision-making contexts under uncertainty.

While the system does not account for all stochastic influences inherent in sports (e.g., in-game momentum, injuries), it offers a robust and interpretable baseline that can be enhanced with additional data sources or modeling strategies.

Future work will focus on incorporating player-level metrics, real-time data streams to further improve both predictive power and system transparency.

References

[1] Nvidia, “What is XGBoost?,” NVIDIA Data Science Glossary, 2024. https://www.nvidia.com/en-us/glossary/xgboost/

[2] Scikit-learn, “scikit-learn: Machine Learning in Python,” Scikit-learn.org, 2024. https://scikit-learn.org/stable/

[3] “Kaggle,” Kaggle.com, 2025.https://www.kaggle.com/competitions/march-machine-learning-mania-2025/overview (accessed May 16, 2025).

[4] V. Kanade, “What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices,” Spiceworks, Apr. 08, 2022. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

[5] "Educative Answers - Trusted Answers to Developer Questions," Educative. https://www.educative.io/answers/classification-using-xgboost-in-python

[6] GeeksforGeeks, "What is High Level Design – Learn System Design," GeeksforGeeks, Feb. 02, 2023. https://www.geeksforgeeks.org/what-is-high-level-design-learn-system-design/

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. of the 22nd ACM SIGKDD, 2016