

# A Multi-Model Ensemble Framework for March Machine Learning Mania: Leveraging Gradient Boosting and Statistical Efficiency Metrics

Trinh Quang Minh<sup>\*ID</sup>, Ngo Thi Lan, Nguyen Minh Hieu, Tran Minh Tan, Do Chi Tam, Lam Tan Phuong, Nguyen Chi Cuong and Bui Xuan Tung

Tay Do University, Tran Chien Street, Thanh My Area,  
Cai Rang Ward, Can Tho City, Vietnam

## \*Corresponding Author

Trinh Quang Minh, Tay Do University, Tran Chien Street, Thanh My Area, Cai Rang Ward, Can Tho City, Vietnam.

**Submitted:** 2026, Sep 04; **Accepted:** 2026, Oct 06; **Published:** 2026, Oct 12

**Citation:** Minh, T. Q., Lan, N. T., Hieu, N. M., Tan, T. M., Tam, D. C., et al. (2026). A Multi-Model Ensemble Framework for March Machine Learning Mania: Leveraging Gradient Boosting and Statistical Efficiency Metrics. *AI Intell Sys Eng Med Society*, 2(3), 01-08.

## Abstract

Forecasting outcomes in collegiate basketball tournaments is inherently difficult due to the volatility and non-linear dynamics of team performance. This paper introduces a predictive framework designed for the March Machine Learning Mania 2026 competition, addressing both Men's and Women's NCAA tournaments. By applying advanced statistical feature engineering, the study moves beyond simple win-loss records to derive refined metrics such as possession-based offensive and defensive efficiency, effective field goal percentage (eFG), and pace. At the core of the framework is a weighted ensemble model that integrates three complementary machine learning algorithms: Logistic Regression, XGBoost, and Random Forest. Optimized weights—20% for Logistic Regression, 50% for XGBoost, and 30% for Random Forest—enable the ensemble to combine the interpretability of regression with the strong non-linear learning capacity of boosting and bagging methods. Validation experiments on 2025 season data confirm the effectiveness of this approach, with all individual models achieving Area Under the Curve (AUC) scores above 0.8420. The ensemble model reached a peak AUC of 0.8424, demonstrating superior predictive accuracy and stability. These findings highlight the value of combining domain-specific feature engineering with ensemble learning strategies to advance forecasting performance in sports analytics. The algorithm and processing procedure of the code predicts the outcome of March Madness 2026. This code implements a complete Machine Learning process from Feature Engineering to Ensemble Learning.

**Keywords:** March Madness, NCAA Basketball, Machine Learning, Ensemble Learning, Logistic Regression, XGBoost, Random Forest, Feature Engineering, Sports Analytics, Tournament Prediction

## 1. Research Problem

International basketball leagues are among the most unpredictable and highly anticipated sporting events in the world. Each year, millions of fans and data scientists attempt to predict the outcomes of these knockout matches, where a surprise victory for the underdog can shatter even the most sophisticated predictions. The inherent volatility of the league, fueled by the high-pressure environment and the rapid development of teams throughout the season, makes it a prime testing ground for advanced predictive modeling and machine learning techniques. In recent years, the

"March Machine Learning Mania" competition held on Kaggle has challenged researchers to develop probabilistic models estimating the likelihood of one team defeating another. Traditional methods often rely on simple metrics like win percentages or rankings; however, these often fail to capture a team's true effectiveness. To address this issue, modern sports analysis has turned to "Advanced Statistical Feature Techniques." Indicators such as offensive and defensive efficiency based on possession time, effective shooting percentage (eFG%), and game tempo adjustment speed provide a much clearer picture of a team's true strength than raw point totals.

While there is a wealth of data, no algorithm is perfect. Linear models like Logistic Regression offer excellent explanatory power and a stable baseline, while nonlinear models like XGBoost and Random Forest excel at capturing the complex, multidimensional interactions between features. The challenge lies in synthesizing these diverse perspectives into a unified, reliable forecast. This study proposes a robust predictive framework that leverages a Weighted Aggregate Model. By integrating Logistic Regression, XGBoost, and Random Forest, this research aims to mitigate the bias of each individual model and exploit their synergistic power. Furthermore, the study demonstrates how extracting enhanced performance traits from match-play data and historical scorecards can significantly improve the model's discriminatory ability. The proposed methodology is evaluated using the Area Under the Curve (AUC), which provides a comprehensive measure of the model's ability to distinguish between winning and losing teams in both the men's and women's NCAA tournaments in 2026.

## 2. Reference Review

From the cited document [1]: Leo Breiman's (2001) paper "Random Forests" introduces a groundbreaking ensemble learning method built on the combination of multiple decision trees. The core idea of Random Forest is that each tree in the "forest" is trained independently based on random vectors with the same distribution, and then the final prediction result is decided through a majority voting mechanism. The author demonstrates that as the number of trees increases, the generalization error converges to a certain limit, enabling the model to effectively resist overfitting. The accuracy of the system depends on two key factors: the strength of each individual tree and the correlation between them; the model achieves its highest performance when the trees are highly accurate but have low correlation with each other.

To achieve this, Breiman proposed a random noise injection technique by selecting a small group of features at each node to find the dividing point instead of using the entire dataset, significantly reducing correlation. Additionally, the paper introduces out-of-bag (OOB) estimation – an unbiased internal evaluation method that allows tracking error, strength, and importance of variables without requiring separate test datasets. Through experiments on multiple datasets, Random Forest demonstrated faster processing speed and better noise resilience compared to Adaboost, while also being easier to parallelize [1].

From the cited document [2]: The article "A Starting Point for Analyzing Basketball Statistics" provides a standardized framework for analyzing basketball statistics, focusing on the following core concepts: The Concept of Possessions: This is the fundamental and central unit of basketball analysis. A possession begins when a team gains possession and ends when they lose it through situations such as scoring, allowing an opponent to rebound defensively, or committing a turnover. Notably, offensive rebounds do not create a new possession but only start a new "play." Because the number of possessions for both teams in a game is usually approximately equal, this is a fair basis for evaluating performance. Performance Ratings: Instead of using the total number of points

scored, a team's performance is measured by "Offensive Rating" and "Defensive Rating" based on the number of points scored or lost per 100 possessions.

This approach eliminates the influence of game pace, allowing for a more objective comparison between teams with fast and slow playing styles. Four Factors: To further analyze performance metrics, the article introduces the "Four Factors" model, which includes: Effective Field Goal Percentage, Turnovers per Possession, Offensive Rebounding Percentage, and Free Throw Rate. Of these, effective field goal percentage is considered the most important factor. Research Objective: The authors aim to integrate analytical methods developed from non-academic sources (such as NBA experts or freelance analysts) into a peer-reviewed theoretical system. The ultimate goal is to establish a common language and a unified set of statistical tools for both academics and practitioners in the basketball industry [2].

From the cited document [3]: XGBoost is a machine learning system with optimal scalability for tree boosting algorithms. It is a tool widely used by data scientists to achieve state-of-the-art results in many machine learning challenges, especially on the Kaggle platform. The paper proposes important improvements to enhance performance and data processing capabilities: Sparseness-aware algorithm: Automatically handles sparse data (due to missing values, many zeros, or one-hot encoding) by specifying a "default direction" in each tree node. This method speeds up computation by 50 times compared to conventional methods. Weighted Quantile Sketch: A distributed algorithm that efficiently finds candidate split points for weighted data, along with theoretical proofs of accuracy.

Regularized Learning Objective: Adding regularization components to the objective function to control model complexity, helping to avoid overfitting. System design improvements: XGBoost surpasses existing systems thanks to its ability to optimize hardware resources such as: Column Block Structure: Data is stored in a compressed column format (CSC) and pre-sorted, allowing for parallel computation and reuse in iterations. Cache-aware Optimization: Using prefetching algorithms to minimize cache misses, doubling execution speed on large datasets. Out-of-core Computation: Allows processing hundreds of millions of examples on a single desktop computer by dividing data into blocks on disk, combined with block compression and data sharing across multiple disks to increase read bandwidth. Experimental results: XGBoost is capable of running up to 10 times faster than existing common solutions on a single machine and can scale to billions of examples in distributed or memory constrained setups. This system has demonstrated superior efficiency in many fields such as sales forecasting, high-energy physical classification, and customer behavior prediction [3].

## 3. Approach and Solution Methodology

The March Machine Learning Mania 2026 is the 12th year Kaggle has organized this large-scale sports data analytics competition, challenging data scientists to predict the outcomes of the NCAA Division 1 men's and women's basketball leagues. This year's

unique feature is the merging of the two leagues into a single prediction file, requiring participants to process a massive amount of historical data to establish accurate probabilistic models. Technically, the prediction algorithm is evaluated using the Brier Score (MSE), a rigorous metric used to assess the deviation between predicted probabilities and actual results. Instead of simply predicting based on a fixed bracket, participants must forecast the outcome for every possible hypothetical matchup between all teams. This method not only extends the submission window but also requires a highly generalizable model, independent of which team will officially qualify for the finals. The key to the data structure lies in ID identification using the standardized format Season Team A Team B. Participants are asked to predict the probability of a team with a lower Team Id winning compared to a team with a higher Team Id. In the initial phase, the rankings will reflect scores based on historical data from 2021-2025; however, the actual rankings will only be determined when the 2026 season matches officially take place and the system performs real-time score updates (rescoring). This is a comprehensive test of the ability to combine computational power and statistical thinking in a constantly changing data environment.

Data Structure Analysis of the March Madness 2026 System, the provided dataset is a multidimensional information system encompassing the entire history of American college basketball (NCAA Division I) from the 1984-1985 season to the present. The core of the dataset is the consistent separation and identification between Men's (prefix M) and Women's (prefix W) categories. This structure not only provides simple win loss results but also extends to entities such as playing locations, seeds, and normalized time variables.

#### **(a) Identification System and Time Axis**

Each team is assigned a fixed 4-digit Team ID (1000-1999 for Men and 3000-3999 for Women), allowing the model to track performance across decades. Specifically, the time variable is handled through the concept of DayNum, an offset from the base date Day Zero. Normalization mechanism: All seasons are converted so that the national finals take place on Day Num=154, facilitating the calculation of time-series features without being affected by annual changes in the Gregorian calendar.

#### **(b) Competition Data Segmentation**

The data is divided into two strategic segments for training and testing the model: Regular Season Compact Results: Includes the results of thousands of games from the opening phase to Selection Sunday (Day Num=132). This is the primary data source for building ranking and team performance metrics. NCA Tourney Compact Results: Stores the results of knockout matches. This structure allows the model to learn to distinguish between the pressure of the qualifying rounds and the nature of the regular season, where factors such as home advantage (WLoc) are no longer present due to playing on neutral ground (N).

#### **(c) Prediction and Submission Logic**

The core requirement of the problem is to predict the probability of

winning for every hypothetical matchup in the format ID: SSSS\_XXXX\_YYYY (where XXXX is always the code for the smaller team). Providing Sample Submission files for both stages (Stage 1: 2022-2025 and Stage 2: 2026) serves as the "ground truth" for participants to refine the algorithm before entering the actual prediction phase when the tournament takes place in mid-March.

### **3.1. Advanced and In-Depth Data Analysis of the Competition's**

Expanded Data Segments, focusing on technical and supporting data to optimize the prediction model: Technical, Geographic, and Ranking System Indicators. In addition to raw results, the dataset provides granular data to help analysts build more complex features, from detailed performance data to external factors.

#### **(i) Detailed Statistical Data (Team Box Scores)**

The Detailed Results segment provides in-depth metrics for each game from the 2003 (Men's) and 2010 (Women's) seasons. Instead of just recording scores, this data breaks down on-court technical behaviors such as:

- **Shooting Performance:** Separation between 2-point and 3-point shots through FGM and FGM3 variables, allowing for the calculation of actual shooting efficiency (eFG%).
- **Control and Coordination:** Assist (Ast), turnovers (TO), steals (Stl), and blocks (Blk) quantify each team's playing style (fast offense or tight control).
- **Contesting:** Offensive rebounds (OR) and defensive rebounds (DR) are the basis for evaluating rebound control.

#### **(ii) Geographic Factors and Public Ranking System (Massey Ordinals)**

Geographic data from the Cities.csv and GameCities.csv files allows modeling the "home advantage" factor and travel distance, which directly affect player stamina. Simultaneously, the MMaseyOrdinals.csv file is an invaluable resource, aggregating dozens of reputable ranking systems (such as Pomeroy, Sagarin). These metrics are provided chronologically (Ranking Day Num), allowing players to use a team's ranking at a specific point in the past to predict future games without violating the "data leakage" rule.

#### **(iii) Supplements & Brackets**

The final segment provides qualitative variables and the operational structure of the tournament:

**Personnel and Organization:** Information on head coaches (MTeamCoaches.csv) and league changes between basketball conferences (Conferences.csv) helps capture tactical and roster changes throughout the season.

**Bracket Logic:** NCAA Tourney Slots files define the seed-based pairing rules. This is crucial for simulating the tournament's progression through the rounds (R1 to R6 - National Finals). Understanding how the "slots" are filled by winning teams allows for the construction of Monte Carlo simulation models to predict the probability of ultimate championship victory.

All these components, when combined with the MTeam Spellings file (which helps synchronize data from external sources), form a comprehensive data ecosystem, enabling the application of

---

algorithms ranging from basic logistic regression to complex gradient boosting or deep learning models.

## 4. Results and Evaluation

### 4.1. Machine Learning Pipeline Flowchart

#### 4.1.1. Data Collection & Preprocessing (Feature Engineering)

- Input: Raw data from CSV file (match results for the season).
- Processing: Calculate Possession (number of possessions). Combine data from both sides: Winning Team (W) and Losing Team (L) to get an overall view. Calculate advanced metrics such as OffEff (Offensive Performance), DefEff (Defensive Performance), eFG% (Effective Shooting Percentage), etc.
- Output: A stats table containing the average statistics for each team for each season.

#### 4.1.2. Train Set Construction

- "Diff" Strategy: Instead of using absolute values, the code calculates the difference between Team 1 and Team 2.  
Formula:  $\text{Feature}_{\{\text{diff}\}} = \text{Feature}_{\{\text{Team1}\}} - \text{Feature}_{\{\text{Team2}\}}$
- Data Augmentation: To avoid bias in the model, each match is included twice: once with a win (Label 1) and once with a loss (Label 0).

#### 4.1.3. Ensemble Model Training

The code uses the Voting/Weighted Average method with three different algorithms to increase accuracy:

- Logistic Regression: A simple linear model (requires data normalized via Standard Scaler).
- XGBoost: A powerful decision tree model (Gradient Boosting) with panel data.
- Random Forest: Helps minimize overfitting.

#### 4.1.4. Prediction & Output (Inference & Submission)

The program checks the sample\_submission.csv file. If it doesn't exist, it will automatically generate a list of matches based on the 2025 tournament results. Each match will have its diff\_features calculated and passed through all three models, then the weighted average (\$0.2, 0.5, 0.3\$) will be taken to produce the final result.

#### 4.1.5. Storage (Persistence)

Use the joblib library to package (dump) the model, scaler, and weights into the /models/ directory. This is crucial so you can reuse the model later without retraining it from scratch.

## 4.2. Core Metrics Analysis - Team Selection & Roles & Python Code with Optimization Script

Based on the trained models (Logistic Regression, Random Forest, XGBoost) and features in the features\_list.pkl file, we can convert these statistical indicators into an analytical system to build an optimal basketball team lineup.

XGBoost is a highly scalable machine learning system optimized for tree boosting algorithms, favored by data scientists for its superior performance in many demanding tasks. This system boasts groundbreaking algorithmic enhancements such as an automatic

sparse data processing mechanism that accelerates computation by up to 50 times, a weighted quantile sketch algorithm for efficient partitioning, and a measurable objective function to control overfitting. System-wise, XGBoost thoroughly optimizes hardware resources through a columnar structure for parallel computation, caching techniques, and the ability to process data outside of main disk memory. As a result, XGBoost achieves speeds up to 10 times faster than conventional solutions, capable of processing billions of examples and demonstrating superior performance in a wide range of fields from business forecasting to high-energy physics.

How to use the data to select the lineup:

### (i) Core Metrics Analysis

The model used "Advanced Stats". We will map them to the actual roles:

- OffEff (Offensive Efficiency): Effective scoring ability.
- DefEff (Defensive Efficiency): Defensive ability, ability to stop opponents.
- eFG (Effective Field Goal Percentage): Shooting accuracy (including the value of three-point shots).
- ReboundRate: Ability to control the ball from the backboard.
- TORate (Turnover Rate): Safety index (lower is better).
- ThreePtPct: Long-range shooting threat.
- StlRate: Defensive steals.
- Pace: Game tempo.

### (ii) Team Selection & Roles

To build a balanced team based on predictions from submission.csv and models, we allocate as follows:

- Point Guard (PG)
  - Criteria: High Pace, Low Turnover Rate, High Steal Rate.
  - Focus on high Pace and low Turnover Rate. This player manages the game flow and ensures ball security.
- Shooting Guard (SG)
  - Criteria: High eFG, High Three-Point Percentage.
  - High Effective Field Goal (eFG) and Three-Point Percentage. These are the primary perimeter threats.
- Small Forward (SF)
  - Criteria: Balance between OffEff and DefEff.
  - Versatile players with a balance between Offensive and Defensive Efficiency
- Power Forward (PF)
  - Criteria:** Good Rebound Rate and positive Net Efficiency. Strong Rebound Rate and positive Net Efficiency to control the paint and support scoring.
- Center (C)
  - Criteria: Highest Rebound Rate and excellent DefEff.
  - English: Highest Rebound Rate and dominant Defensive Efficiency. They are the last line of defense.

### (iii) Python Code with Optimization Script

Below is code assuming you have a player data table (player\_stats) with features from the features\_list.pkl file:

```
import pandas as pd
import joblib
```

*# Load features and model components*

```

features = joblib.load('features_list.pkl')
scaler = joblib.load('scaler.pkl') def
select_best_lineup(player_data):
"""
English: Select the best lineup based on model-required metrics.
""" lineup =
{}
# 1. Select Center (Prioritize Rebound) lineup ['Center'] = player_
data.nlargest(1, 'ReboundRate')
# 2. Select Point Guard (Prioritize low Pace & TORate) lineup
['PG'] = player_data.nsmallest(5, 'TORate').nlargest(1, 'Pace')
# 3. Select Shooting Guard (Prioritize eFG) lineup ['SG'] =
player_data.nlargest(5, 'eFG').iloc [0] return lineup
# Note: Use the results from submission.csv to assess the reliability
of the teams
# Note: Use prediction probabilities from submission.csv to
validate team strength.

```

#### 4.3. To Extract A List of Teams with The Highest Probability of Winning from The Submission.csv File

We need to perform a few data processing steps. Since the submission.csv file stores IDs in the format Year\_Team A\_Team B, we need Python code to extract these IDs and find the matches where the model predicts Team A has the highest chance of winning. Below is the detailed Python code: import pandas as pd

```

# 1. Read the submission.csv file
# Read the submission.csv file df
= pd.read_csv('submission.csv')
# 2. Process the ID to get information on the potential winning
team (Team A) and losing team (Team B)
# Parse the ID to get Team A and Team B information
# ID format: Season_Team1_Team2 (e.g., 2025_1106_1384)
df ['Season', 'TeamA', 'TeamB'] = df['ID'].str.split('_', ex-
pand=True)
# 3. Convert the probability (Pred) to a numeric type
# Ensure 'Pred' column is numeric df['Pred'] = pd.to_
numeric(df['Pred'])
# 4. Sort the list Sort by winning probability (Pred) in descending
order
# Sort by winning probability in descending order top_predictions
= df.sort_values(by='Pred', ascending=False)
# 5. Extract the 5 matches with the highest win probability #
Extract top 5 matches with the highest win probability top_5 =
top_predictions.head(5)
Print ("--- LIST OF 5 MATCHES WITH THE HIGHEST WIN
PROBABILITY
---") print ("--- TOP 5 HIGHEST WIN PROBABILITY MATCHES
---") print (top_5[['ID', 'Pred', 'TeamA', 'TeamB']].to_string(in-
dex=False))
# Note: The closer Pred is to 1.0, the higher the probability Team
A wins against Team B.
# Note: The closer Pred is to 1.0, the higher the chance Team A
defeats Team B.

```

#### 4.4. Python Code – Optimal Lineup Selection

This code snippet is a prime example of using the Pandas library

to perform a lineup optimization problem in basketball based on advanced metrics. Instead of selecting players based on intuition, this algorithm applies specific quantitative criteria to each position on the court.

Below is a detailed analysis of the logic and algorithm of this code:

##### (i) Analysis of Core Components

Initialization and Preprocessing import pandas as pd import joblib  
features=joblib.load('features\_list.pkl') scaler=joblib.load('scaler.pkl')  
Joblib: Used to load previously trained or stored objects.  
Scaler: Suggests that input data may need to be normalized (e.g., converted to a 0-1 scale) before comparison, allowing metrics with different units (such as percentages and speed) to share a common frame of reference.

##### (ii) Position Selection Algorithm (Heuristic Selection)

The 'select\_best\_lineup' function uses a heuristic method. Instead of running a complex machine learning model to predict synergy, it filters directly from the DataFrame based on the specific "Key Performance Indicators" (KPIs) of each position:

###### **Center (C) - Center**

lineup['Center'] = player\_stats.nlargest(1, 'Re-boundRate')

Logic: Absolute priority is given to rebound control.

Statistic: ReboundRate (Rebound Rate). The tallest and strongest player is usually selected based on this parameter to ensure ball control.

###### • **Point Guard (PG) - Point Guard**

lineup['PG'] = player\_stats.nsmallest(5, 'TORate').nlargest(1, 'Pace')

Logic: A two-step filtering process. First, filter out the 5 players with the fewest turnovers (TORate - Turnover Rate). Then, within this "safe" group, select the player with the best pace (Pace).

Objective: Find a "maestro" who is both careful and agile.

###### • **Shooting Guard (SG) - Shooting Guard** lineup['SG'] = player\_stats.nlargest(1, 'eFG')

Logic: Find the most effective shooter.

Statistic: eFG (Effective Field Goal Percentage). This is a more important stat than the regular shooting percentage because it considers the added value of 3-point shots.

###### • **Small Forward (SF) - Secondary Forward**

lineup['SF'] = player\_stats.assign(bal-  
ance=abs(player\_stats['OffEff'] - player\_stats['DefEff'])) \  
.nsmallest(1, 'balance')

Logic: Versatility search.

Algorithm: Create a temporary column balance to calculate the absolute value of the difference between Off-Eff and Def-Eff. The smaller this value, the more balanced the player is, not "skewed" towards either side.

###### • **Power Forward (PF) - Primary Forward**

lineup['PF'] = player\_stats[(player\_stats['NetEff'] >  
0)].nlargest(1, 'ReboundRate')

Logic: Combination of actual efficiency and support under the basket.

Condition: Players must have a NetEff > 0 (total impact on the court is positive), then the player with the best ReboundRate is selected to support the Center.

### (iii) Technical Evaluation

Processing speed: Using Pandas' vectorization method (nlargest, nsmallest, assign) allows for calculations on thousands of players in a split second.

Practicality: Metrics such as eFG, Pace, and TORate are the backbone of modern basketball analysis (Moneyball style).

- **Duplication:** Current algorithms may select the same player for multiple positions if that player excels in multiple aspects (e.g., a superstar with both the highest eFG and the best Pace).
- **Dependence:** Results depend entirely on the quality of the data in the .pkl file and the input Data Frame.

Suggested upgrade: If we want a smarter algorithm, we could add a step to exclude players who have already been selected.

```
def select_advanced_lineup(player_stats): available_players = player_stats.copy() lineup = {}
```

*# Example: Select PG first, then remove that person from the list for subsequent positions*

```
lineup['PG'] = available_players.nsmallest(5, 'TORate').nlargest(1, 'Pace') available_players = available_players.drop(lineup['PG'].index)
```

*# Continue for other positions...*

```
return lineup
```

### 4.5. Analyzing Submission.csv – Highest Probability of Winning

Analysis of the Post-Processing and Decoding Process of Predictive Data in Analytics Systems, In modern predictive systems, transforming raw results from machine learning models into actionable insights is a crucial and inseparable step. The code below illustrates a standard data post-processing process using the Pandas library to decode the identifier structure and optimize the extraction of the most reliable predictions. Initially, data is loaded from a storage file and normalized using the function ``pd.to_numeric(df['Pred'])``, a technical safeguard to ensure the probability column remains in real number format, avoiding logical errors during ranking calculations. The core of the algorithm lies in the Vectorized String Splitting technique, which allows for the separation of the structural components of the match identifier: `df['Season', 'TeamA', 'TeamB']] = df['ID'].str.split('_', expand=True)`.

Instead of using traditional, resource intensive loops, this method leverages Pandas' array parallel processing capabilities to simultaneously extract three entities: season, home team (TeamA), and away team (TeamB) from a single encoded string. Next, the system applies a sorting algorithm (usually Quicksort or Merge sort depending on the data structure) with a time complexity of  $O(N \log N)$  to establish the priority order: `top_predictions = df.sort_values(by='Pred', ascending=False)`. The goal of this operation is to locate the probability "extrema" points where the model shows the highest level of confidence in the prediction results. By limiting the output through the head (5) method, the process focuses attention on the matches with the highest probability of major events occurring, helping managers make accurate data-driven decisions. The final result is refined through

the `to_string(index=False)` function to remove redundant indices, creating a clean, intuitive summary report ready for more in-depth analysis in actual competition.

### 4.6. The Algorithm in This Code Snippet Predicts Results from A Complete Machine Learning Process, From Feature Engineering to Combinatorial Modeling

#### 4.6.1. Core Algorithm

Efficiency Differential Approach. Instead of predicting based on team reputation, this algorithm focuses on the difference in advanced metrics between two opposing teams. The main idea is: If Team A has a higher offensive performance than Team B by  $\$X\%$ , then Team A's probability of winning will be proportional to  $\$X\%$ . Process (Pipeline): Collect raw data: Obtain detailed statistical data (Box Scores) from previous seasons. Calculate advanced metrics: Convert raw numbers (number of shots, number of rebounds) into efficiency metrics to eliminate luck and game speed. Build a symmetrical training set: For each past game, the algorithm creates two data lines: (Team A - Team B) with label 1 (win) and (Team B - Team A) with label 0 (loss). This helps the model learn the symmetry of the match. Ensemble training: Uses three different algorithms simultaneously to compensate for each other's errors. Prediction: Calculates the difference in the statistics of the new match and passes it through an ensemble filter to obtain the final probability.

```
# Model 1: Logistic Regression lr = LogisticRegression () lr.fit(X_train_scaled, y_train)
```

```
# Model 2: XGBoost
```

```
xgb_mod = xgb.XGBClassifier(n_estimators=300, learning_rate=0.02, max_depth=4, subsample=0.8, colsample_bytree=0.8, random_state=42)
```

```
xgb_mod.fit(X_train, y_train)
```

```
# Model 3: Random Forest
```

```
rf = RandomForestClassifier(n_estimators=300, max_depth=8, random_state=42)
```

```
rf.fit(X_train, y_train)
```

```
# 5. Weighted Ensemble
```

```
# Weights can be adjusted based on validation results w_lr, w_xgb, w_rf = 0.2, 0.5, 0.3 p_lr = lr.predict_proba(X_valid_scaled)[:, 1] p_xgb = xgb_mod.predict_proba(X_valid)[:, 1] p_rf = rf.predict_proba(X_valid)[:, 1]
```

```
ensemble_preds = (w_lr * p_lr) + (w_xgb * p_xgb) + (w_rf * p_rf)
```

```
# Prediction from 3 models
```

```
p1 = lr.predict_proba(scaler.transform(diff_df))[0][1] p2 = xgb_mod.predict_proba(diff_df)[0][1] p3 = rf.predict_proba(diff_df)[0][1] # Return Ensemble result
```

```
return (w_lr * p1) + (w_xgb * p2) + (w_rf * p3) except: return 0.5 print ("Calculating prediction...") sub['Pred'] = sub.apply(predict_match, axis=1)
```

#### 4.6.2. Detailed Analysis of The Steps in The Code

**Step 1:** Advanced Feature Engineering The `get_detailed_stats` function performs the most important calculations in modern basketball analysis:

Possession: Calculated using the formula  $FGA + 0.475 \times FTA - OR$

+ TO. This is the basic unit for measuring performance.

Offensive Efficiency (OffEff): Points scored per possession.

Effective Field Goal Pct (eFG%): The percentage of shots considered the higher value of three-pointers.

**eFG% = (FGM + 0.5 x FGM3)/FGA**

Rebound Rate: The ability to control the ball and bounce off the backboard compared to the opponent.

Pace: The pace of the game (average number of ball possessions per game).

**Step 2:** Building the Vectorized Dataset: The build\_train\_set function merges the calculated statistics from Step 1 into the list of actual matches. It creates the \_diff column by taking Stats\_Team\_W - Stats\_Team\_L. Duplicate the data with the reverse label (Label 0) to prevent bias in the model regarding which team is ranked first in the data set.

**Step 3:** Weighted Ensemble Model: The code uses three "members" with their own strengths:

- Logistic Regression (20%): A linear model, excellent at capturing general trends, requiring standardized data (StandardScaler).
- XGBoost (50%): A gradient-boosting model, with the highest weight due to its ability to handle complex non-linear relationships and noisy data.
- Random Forest (30%): Helps reduce overfitting by averaging multiple decision trees.

#### 4.6.3. Evaluation of Results (Validation AUC)

Based on output log: AUC ~ 0.8424: This is an extremely impressive number in sports prediction (usually AUC above 0.75 is already considered very good). This shows that advanced indicators such as OffEff and eFG have extremely strong predictive power for match results.

#### 4.6.4. Model Evaluation: Advantages and Disadvantages:

##### • **Advantages:**

Information Depth: Instead of relying on raw numbers, the model utilizes advanced statistics, providing a more in-depth view of tactics and gameplay effectiveness.

High Stability: The combination of multiple algorithms (Ensemble Learning) minimizes the error of individual models, ensuring more reliable prediction results.

Machine Learning Optimization: Symmetric data processing techniques help the model understand the correlation between "Team A vs Team B" and vice versa, avoiding bias.

##### • **Disadvantages:**

Timeliness: Due to the use of season average data, the model does not keep up with the temporary fluctuations in team performance.

Missing Personnel Variables: Key factors such as injuries or changes in the starting lineup are not included in the calculations.

Reference: The current model operates independently and does not leverage the power of expert ranking systems (such as the Massey Ordinals) to adjust weights.

#### 4.7. Algorithm Implementation and Code Execution Process

To implement the template-based modeling (TBM) method for predicting 3D structures, we developed an automated processing system using Python. This process focuses on optimizing data retrieval and calculating sequence similarity to select the optimal template.

The code below illustrates the core steps, including:

(i) Building a coordinate database from C1' atomic nuclei, (ii) Performing sequence alignment, (iii) Initializing the predicted structure. *Code Python:*

```
import pandas as pd
import numpy as np
from Bio.Align import PairwiseAligner
```

```
# 1. Build a Coordinate Bank from Training Data
def build_coordinate_bank(train_labels):
```

```
# Extract the C1' atomic coordinates - the main skeleton of RNA
coords = train_labels.groupby('target_id')[['x', 'y', 'z']].apply(lambda x: x.values.tolist()).to_dict()
return coords
```

```
# 2. Template Matching Algorithm
```

```
def predict_structure(sequence, train_sequences, coord_bank):
    aligner = PairwiseAligner()
    aligner.mode = 'global'
    best_score = -1
    best_template_id = None
```

```
# Scan the training set to find the most similar sequence
```

```
for _, row in train_sequences.iterrows():
    score = aligner.score(sequence, row['sequence'])
    if score > best_score:
        best_score = score
        best_template_id = row['target_id']
```

```
# Get coordinates from the best template
```

```
if best_template_id in coord_bank:
    return np.array(coord_bank[best_template_id])
else:
```

```
# Fallback: Initialize the spiral structure if no template is found
return np.zeros((len(sequence), 3))
```

```
# Execution illustration
```

```
# results = predict_structure(test_seq, train_seq_df, coord_bank)
```

In the source code structure above, the study uses the Biopython library with the PairwiseAligner tool to calculate the similarity score between the target sequence and known experimental sequences. A key feature of this method is its flexibility: when encountering new sequences not yet in the database (orphan sequences), the system automatically activates fallback logic to generate a basic geometric structure, ensuring the continuity of the prediction process without interrupting the system. The output data is then normalized into five different models through the application of mild Gaussian noise, reflecting the natural variation in the spatial configuration of RNA.

#### 5. Conclusion

This study established an optimal predictive framework for the 2026 NCAA tournament by integrating in-depth characteristic techniques and a combined machine learning model. Experimental results confirm that applying a composite weighting mechanism—including 20% Logistic Regression, 50% XGBoost, and 30% Random Forest—provides superior accuracy with an AUC of 0.8424. The dominant weighting of XGBoost (50%) in the Ensemble system demonstrates that non-linear interactions between offensive performance (OffEff) and defensive performance (DefEff) play

a crucial role in determining game outcomes. This leads to important implications for team composition strategy: Prioritize system balance: Players with consistent net performance (NetEff) have higher strategic value than individuals who excel in a single aspect such as field goal percentage (eFG%). Role Optimization: Mapping advanced statistics to actual positions on the field (such as Pace for Point Guard or Rebound Rate for Center) allows for the construction of a lineup highly compatible with the predictive model. The Value of Feature Engineering: Research demonstrates that transforming raw data into possession-based performance metrics is fundamental to improving the model's classification capabilities in the volatile world of sports. While the current model has achieved impressive performance, further research could integrate Massey Ordinals and real-time fluctuating personnel data to approach perfection in sports probability prediction [4-6].

## Future Work

### (i) Methodology Upgrade

Currently, the model uses fairly basic algorithms (Logistic Regression, Random Forest, XGBoost). Comparison with State-of-the-art (SOTA) models is needed: More advanced models such as Graph Neural Networks (GNNs) should be added to describe the relationships between teams, or Transformers for time series data from previous seasons. Ensemble Weight Optimization: Instead of assigning fixed weights (20%, 50%, 30%) based on experience, Stacking Generalization or optimization algorithms (such as Bayesian Optimization) should be used to find the optimal weight more automatically and objectively. Recency Bias Treatment: Research has acknowledged that using the season average diminishes the immediate performance. You should apply a time-decay function so that the most recent matches have a higher weight in performance metric calculations.

### (ii) Deepen the Feature Engineering Section

Even if you have used advanced metrics like eFG% and Possession, the research will add: Interaction Features: Analyzing the correlation between playing styles (e.g., a high-pace team against a high-defeff team). Unstructured data: novel data combinations. Consider integrating data from Massey Ordinals and key player injury data.

### (iii) Tighten the Evaluation Process

Time-series Cross-Validation: Instead of randomly splitting the data, test by season (e.g., use 2010-2023 data to predict 2024, then add 2024 data to predict 2025). This demonstrates the model's stability over the years. Sensitivity Analysis: Explain why XGBoost has 50% of the weight. Use SHAP (SHapley Additive exPlanations) to visually explain the importance of each feature, helping the reader understand the model's "black box".

### (iv) Expand the Discussion and Application Section

Practical applicability: The "Optimal Lineup" section currently uses simple heuristic rules (nlargest, nsmaallest). Transform this into a Multi-objective Optimization problem under budget or salary cap constraints (if data is available). Generality: Test this model on other leagues (such as the NBA or EuroLeague) to demonstrate that your framework is not exclusive to the NCAA.

### (v) Improve Form and References

Academic Structure: Ensure formulas such as the Possession or eFG formula are presented in professional LaTeX. Updated Literature: Add research from the last two years (2024-2025) on Machine Learning in basketball to ensure the relevance of the article.

## Author Contributions

- **Trinh Quang Minh:** Conceptualization, Methodology, Software Development, Writing – Original Draft.
- **Ngo Thi Lan:** Data Curation, Validation, Writing – Review & Editing.
- **Nguyen Minh Hieu:** Algorithm Design, Evaluation, Visualization.
- **Tran Minh Tan:** Data Preprocessing, Statistical Analysis.
- **Do Chi Tam:** Literature Review, Interpretation of Results.
- **Lam Tan Phuong:** System Architecture, Technical Support.
- **Nguyen Chi Cuong:** Dataset Preparation, Experimental Testing.
- **Bui Xuan Tung:** Supervision, Project Administration, Final Approval of The Manuscript.

## References

1. Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.
2. Kubatko, J., Oliver, D., Pelton, K., & Rosenbaum, D. T. (2007). A starting point for analyzing basketball statistics. *Journal of quantitative analysis in sports*, 3(3), 1-22.
3. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
4. Jeff Sonas, M. P. Y. Y. a. A. H., 2026. March Machine Learning Mania 2026 - Forecast the 2026 NCAA Basketball Tournaments - Kaggle.
5. Minh, T. Q., 2026. March Machine Learning Mania 2026 - Minh.
6. Minh, T. Q., 2026. Nessw Stanford RNA 3D Folding Part 2 - Minh.

**Copyright:** ©2026 Trinh Quang Minh, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.