Logan Laszewski - March Madness Project March 2025 - Diary

Datasets
https://www.kaggle.com/competitions/march-machine-learning-mania-2025/overview
m_TourneySeeds = pd.read_csv("MNCAATourneySeeds.csv")
m_Teams = pd.read_csv("MTeams.csv")
m_RegCompact = pd.read_csv("MRegularSeasonCompactResults.csv")
m_RegDetailed = pd.read_csv("MRegularSeasonDetailedResults.csv")
m_Seasons = pd.read_csv("MSeasons.csv")
m_Conferences = pd.read_csv("MTeamConferences.csv")
m_TourneyCompact = pd.read_csv("MNCAATourneyCompactResults.csv")

March 16, 2025 - Find Data and Observe

Selection Sunday! (When the bracket is announced)
I used Kaggle to locate the data needed for this project. I found Kaggle's March Mania Challenge, which had numerous datasets available. After exploring the datasets, I selected seven to eight that would be most useful for this project. I decided to follow the competition guidelines and work toward generating a prediction (probability) of a win or loss for each matchup. Additionally, I conducted research on key variables that influence March Madness victories and explored potential models for predictions.

March 17, 2025 - Data Cleaning and Transformation

I began coding in Python. I extracted each unique TeamID from the m_Teams dataset and merged it with the m_RegDetailed dataset, which contains detailed statistics for each regular season game.

Key columns I worked with include:

WScore, LScore
WFGM, WFGA, LFGM, LFGA
WFGM3, WFGA3, LFGM3, LFGA3
WFTM, WFTA, LFTM, LFTA
WOR, WDR, LOR, LDR
WAst, WTO, WStl, WBlk, WPF
LAst, LTO, LStl, LBlk, LPF

From there, I selected variables I found most relevant, including:

Three-point shots were attempted and made
Points scored by and against the team
Turnovers
Offensive rebounds
Personal fouls
Field goals attempted and made

I created a for-loop to calculate the sum of these statistics for each team and season. My new dataset includes each team's aggregated season stats. Additionally, for every team, I included statistics from both their perspective and their opponents' perspective (e.g., Duke's three-point attempts vs. three-point attempts against Duke).

March 18, 2025 - Additional Variables

Next, I created additional variables to enhance the model's predictive power.
Games Played: I added a column to calculate the number of games played for each team. This allowed me to convert cumulative statistics into averages.

Performance Metrics: Calculated metrics like three-point percentage, point differential, effective field goal percentage (eFG%), and free throw rate (free throw attempts / shot attempts).

Seeding Cleanup: I cleaned the seeding variable by removing unnecessary formatting.
Conference Strength: I converted conference information into a strength rating (1-4) based on rankings to use as a model feature.

Note: I removed the ordinal ranking column after finding it unhelpful for my model.

March 19, 2025 - Model Building

With the data prepared, I began constructing my model. Using the m_TourneyCompact dataset, which contains March Madness matchups and outcomes, I set up the data for model training.

Data Split: Seasons from 2003 to 2020 served as training data, while Seasons from 2021 to 2024 were used as test data.

Predictors: Over 50 predictors were included using both team and opponent statistics.
Model Choice: I implemented an XGBoost model.

Initial Results: Training Accuracy: ~90% Test Accuracy: ~63%
Log Loss and AUC Scores: Mediocre
The performance indicated a possible overfitting issue.

March 20, 2025 - Model Improvement

To address the overfitting problem and improve model performance, I applied several techniques:

GridSearch: I used GridSearch to tune hyperparameters and optimize for the best AUC and log loss scores.

Recursive Feature Elimination (RFE): Implemented RFE to select the most important predictors, reducing unnecessary complexity and mitigating overfitting.

Explored Alternative Models: I experimented with models using the H2O package but found no significant performance improvements over XGBoost.

The best results with adjustments: Test Accuracy: ~69% Train Accuracy: ~71%

March 21, 2025 - Predicting 2025 Games and Observing Probabilities

The first round has begun!

Using the m_TourneySlots dataset, I extracted the matchups for the 2025 tournament and merged them with the team stats dataset. This prepared the data for input into the model.

Probability Generation: I produced a list of predicted probabilities for each matchup.
Model Insights: I analyzed feature importance visualizations to understand how predictions were made.

Findings:

Seed Overreliance: The model heavily weighted seeding, often selecting the higher seed without capturing potential upsets.

Alternative Metrics Issue: After removing the seeding feature, point differential and team wins became overly dominant.

Next Steps:
If I were to continue this project, I would investigate additional predictors that may help identify underdog victories. Ideas include:

Pace of play
Coaching experience
Recent win percentage (e.g., performance in the last month)
This project has given me valuable experience in data collection, cleaning, feature engineering, model building, and evaluation. I also gained insights into balancing model accuracy and interpretability while understanding limitations in predictive modeling for unpredictable events like March Madness.

Next Steps

I will continue on working on how to improve my model and figure out the current issues I am running into.

June 2025, Improvements and wrapping up project

To avoid models that simply reinforce picking favorites, I intentionally removed variables that make outcomes too predictable—such as point differential, win count, conference rank, and seeding. The goal was to build models capable of making bold, data-driven upset predictions, not just reaffirming conventional wisdom.

I also simplified the feature set by converting all "for" and "against" stats into margin-based metrics. First, I calculated averages for each team's "for" and "against" values, then created margin stats by subtracting the two. This approach reduced feature redundancy and improved model clarity.

Final Model Iterations:

XGBoost: The latest version achieved 70% training accuracy and 60% test accuracy—lower than earlier models but more in line with our goal of avoiding obvious predictors and encouraging underdog picks.

H2O AutoML: I used H2O's AutoML to compare models like GLM, Random Forest, and GBM. The top performer was a GLM (logistic regression) with ~71% train accuracy and ~69% test accuracy. These are strong results given that the model does not use win totals, seeds, or rankings—showing it can still uncover meaningful patterns.

Feature Improvements: I added margins for assists, rebounds, steals, blocks, and possessions to better capture differences in team playstyles and effectiveness beyond scoring.