Examining the Relationship Between Player Archetypes and Winning in College Basketball

DSC 394 Senior Capstone

Christian Rosse

Capstone Advisor: David Hubbard

Abstract

When two college basketball (CBB) programs face off, what factors determine the winner? With the growing popularity and expansion of CBB at the division-one level, more coaching staffs seek out answers to gain even the slightest edge, because there are thousands if not millions of dollars at stake. This study uses granular level box score data on a player level spanning from Nov. 2022 to Jan. 2026 and features a select mix of high-major, mid-major, and low-major conferences. This study leveraged K-Means clustering techniques to create archetype labels to develop matchup level relational analysis in attempts to grasp the nature of how coaches design winning teams against their competitors.

With the data trained on the 2022-2026 seasons and 12,474 individual D-I CBB games, an Extreme Gradient Boosting model was developed and tuned using Calibrated Classifier Cross Validation technique to predict outcomes of CBB games with an accuracy of 73% and an AUC of 0.812. With results indicating a significantly higher accuracy than a 50-50-coin flip, an underlying nature within matchup structures can be inferred and explored further. A larger breadth of training data, exploration of number of archetypes and model accuracy, and weight tuning can provide a list of future steps to ensure continued model improvement.

1. Introduction

Every year, more than 360 universities across the United States put their best talent forward during the five-month span of November to March with one goal in mind: To find themselves in the “March Madness”

(MM) tournament. The tournament brings many eyes upon previously unknown programs buried within the middle of the 360, helps players showcase their talents for professional transition, and provides athletic programs with thousands of dollars for their athletic departments. With so many universities competing in games every day during the season, there is a large breadth of in-game data being generated every month games are held. This breadth of data becomes necessary information for fans, media members, and commentators when tasked to effectively assess the product they consume. But most importantly this data can be used for coaching staffs to evaluate talent and optimally set their team up for the most success to make a run for the championship once March rolls around.

The journey to MM is begun in the offseason where the role of athletic directors and head coaches has expanded greatly over the last few years due to the expansion of the transfer portal (TP). The TP represents a pipeline that allows college athletes to address their intentions to transfer programs. With the expansion of Name-Image-Likeness (NIL) payment, and revenue sharing, allowing players to get paid their value amongst a free market, the TP offers a life-changing opportunity for players to seek roles at larger institutions to not only improve their development as a player, but also seek higher financial compensation while doing so. With only five starting positions in a lineup, basketball is systematically susceptible to high turnover rates year-to-year from the TP. In the offseason in 2025, over 2,300 players entered the transfer portal with intent to change teams (Salao, 2025). With so much turnover, and hundreds of thousands being invested into new players, programs need security to make the best decisions possible in the offseason, which will result in the main goal during the season: Wins.

Winning comes in many different ways as variance is inherently prevalent in the sport of basketball. Mere inches can decide whether a shot is made or missed. Referees make subjective calls which influence individuals' playing time, final scores, and strategies, and adaptive coaching against this subjectivity and variance is executed on the fly in real time. Examining box score counting trends such as points, rebounds, assists and turnovers leaves blind spots where elements of the game that aren't represented in the stat sheet go unnoticed. Two players can score the same number of points, but in vastly different ways. Two players can generate the same number of steals but with vastly different body compositions. Many outcomes defy the linearity of simple home/away, win/loss record, and point based prediction. An underlying factor present includes the nature of five different sets of strengths and weaknesses meshing against one another and tangling with every defensive switch. This presents an ongoing challenge of trying to find ways to represent many of these blind spots in a numerical way in the hopes for more effective predictive modeling. This research aims to grapple with this particular issue in pursuit to effectively account for these matchup combinations, in hopes to provide coaches and fans with tools to get a grasp of how a new team of transfers, or a injury riddled squad, may perform in an unseen matchup.

2. Literature Review

With the expansion of basketball analytics, statistical and machine learning strategies to evaluate the game are becoming increasingly used. This includes the idea of utilizing unsupervised clustering to create player archetypes. This study gained inspiration from Luke S. J. Penner's approach to evaluate archetype groups to optimize roster composition (Penner, 2025). Penner utilized data spanning across a multitude of

basketball leagues and levels, including NBA, NCAA (Divisions I-III), and European professional leagues. Clustering was executed using the K-Means algorithm and evaluated using the Jaccard Coefficient (JC):

$$Jaccard\ Coefficient = \frac{a}{a+b+c}$$

The machine learning yielded nine optimal clusters, and the JC for nine clusters yielded a similarity of 11.89% indicating substantial seperation between clusters.

Individual, unique regression models were trained respective to each individual cluster with points per 48 minutes played (PTS/48) as the target variable. This aspect of the analysis pipeline focuses on the points as the driver for team success since the team with the most points wins the games. Cluster labels were assigned to successful NBA teams and olympic-level foreign teams, and optimal archetype distribution was assessed in a retrospective way instead of prospective.

Penner's work can be expanded elementally as relational analysis was never executed prospectively or predictively. There was no predictive assessment analyzing if a team were to copy a successful archetype-based roster distribution such as the 2019 Warriors, would they have the same success? How might they fall short. Our study aims to expand upon the archetyping process by training machine learning models on lineup archetype composition and developing accurate predictions of in-game success.

3. Methodology

Problem Definition

With the goal in mind to understand the matchup level relationships in a data driven scale, the issue at hand is deriving matchup level metrics out of traditional box score data for modeling. As box score data leaves many blind spots to the human element of in-game matchups, we must engineer a numeric feature that is descriptive to a player's playing style in order to derive unexamined relationships. The feature needed is an archetype label to numerically describe a player's playstyle to aid with machine learning. Player archetyping can provide a scalable solution to expand modeling to new players on new teams addressing the high turnover nature of collegiate basketball.

As ties are impossible given the nature of NCAA rules, our outcome variable 'Win' is represented as a binary (1/0) variable where if $team_i$ wins, win_i will be 1, and 0 if otherwise.

Data Wrangling

The data used for this project was sourced through AccessElias, an online SQL frontend tool that allows querying CBB data to find player trends and statistics. The dataset acquired features single-game player box score results containing typical box score statistics like points (PTS), rebounds (REB), assists (AST), turnovers (TO), blocks (BLK), and steals (STL). The dataset used, features player results from all games played from Nov. 1, 2022, to Jan. 20, 2026, within the following conferences:

- ➔ Southeastern Conference (SEC) - High Major
- ➔ Big EAST Conference (BE) - High Major
- ➔ Big 10 Conference (B10) - High Major

- ➔ Sun Belt Conference (SBC) - Mid Major
- ➔ Mountain West Conference (MWC) - Mid Major
- ➔ Coastal Athletic Association (CAA) - Mid Major
- ➔ Missouri Valley Conference (MVC) - Mid Major
- ➔ Big West Conference (BW) - Mid Major
- ➔ Mid-American Conference (MAC) - Mid Major
- ➔ American Athletic Conference (AAC) - Mid Major

- ➔ Horizon League (HL) - Low Major
- ➔ Northeast Conference (NEC) - Low Major
- ➔ Patriot League (PL) - Low Major
- ➔ Ivy League (IVY) - Low Major
- ➔ Mid-Eastern Athletic Conference (MEAC) - Low Major
- ➔ Western Athletic Conference (WAC) - Low Major

With the interest of time, acquiring data from every conference was deemed unnecessary, and inclusion of a healthy mix of high major, mid major, and low major teams was deemed sufficient for this exploration.

Player characteristics such as height (FF-II), weight (pounds) and position were acquired for all NCAA Division-I CBB players from 2022-2026 through a public sports database website RealGM.com. Height was converted to inches, and positions were number labeled:

Forward (F)---------- 1
Center (C)----------2
Point Guard (PG)----------3
Guard/Forward (GF)----------4
Power Forward (PF)---------- 5
Small Forward (SF)----------6
Guard (G)----------7

There were select players whose position was listed as something other than the seven above. Due to these inconsistencies, label reassignment yielded nulls for these situations. To solve this, we created a supervised K-Nearest Neighbors (KNN) pipeline, assigning the number of neighbors as 15, to fill out the rest of the null labels. Model performance was assessed using cross validation (CV) where n folds were assigned as 5. The KNN model performed a CV accuracy of 0.671 indicating a sufficient performance for multi-class labeling.

Similarly, weight was null for many players from the player characteristics dataset. To solve this, a K-Neighbors Regressor was developed to assign weight values with a similar strategy. The number of neighbors

was set to 5, and the model was evaluated through diagnostic tests of random variance (fig 1) and normality of the residuals (fig 2). With the regressor passing the diagnostic tests, null weight values were filled based on model predictions.

The player characteristic dataset was merged onto the game-by-game result dataset to give every player's performance a respective weight, height, and position. There were inconsistencies in player names across datasets. To remedy this issue, the fuzzywuzzy package in Python utilizes Levenshtein distances to execute fuzzy string matchup and suggest players with stark similarities across the two datasets. Using this library cut the data cleaning and merging process into a fraction of the time.

A few features were added in the interest of exploration. Raw Impact Score (RIS) was created to encompass a player's statistical contribution with one number.

$$RIS = \frac{PTS}{MIN} + 0.7\left(\frac{REB}{MIN} + \frac{AST}{MIN}\right) + 1.5\left(\frac{STL}{MIN} + \frac{BLK}{MIN}\right) - \frac{TO}{MIN}$$

For all players for a given team who played on the same date, the RIS was averaged for each to create a Team Average RIS (TARIS).

$$TeamAvgRIS = \frac{1}{N}\sum_{i=1}^{N} RIS_i$$

Value added (VA) was calculated for each player by finding the difference between the player RIS and the TARIS to create a zero-sum metric for how each player performed against their team's mean in attempts to carve out a representation for a player's overall role.

$$VA_i = RIS_i - GameRIS$$

The last engineered feature was a single-game, implied delta win probability (WP) for each player. This metric roughly examines the fraction of the game's points the player was on the floor for, against the fraction of opponents points the player was on the floor for. This is then applied to a simple probit probability curve respective of point differential and time remaining. For the sake of efficiency, we decided to use a probability model developed by GitHub user anpatton. This new metric represents implied change in win probability the player influenced. This is implied as the exact times the players entered and exited from the on-floor lineup are unknown, so this stands as a rough estimate.

Probit Model (anpatton, 2021)

$$WP(t, H, A, F_h, F_a) = \frac{1}{1 + e^{-(0.7441 - 0.0001264t + 0.1491H - 0.1613A - 0.01901F_h + 0.06214F_a)}}$$

Delta Win Probibility

$$\Delta WP_i = \frac{1}{1 + e^{-(0.7441 - 0.0001264t_i + 0.1491H_i - 0.1613A_i)}} - \frac{1}{1 + e^{-0.7441}}$$

Clustering Model Development

For development of unsupervised cluster labeling, players' points, rebounds, assists, steals, blocks, free throws, and three pointers were aggregated to a per minute basis. This was done to scale players who play a low number of minutes to be assessed and clustered relatively to what they offer when on the floor. The average VA for each player was also calculated across the games they recorded statistics for. Lastly, player frame characteristics height (inches) and weight (pounds) were added as well.

To determine the optimal number of clusters, K-Means models were created with k-values 2-11 for optimal cluster diagnosis. An elbow plot (fig 3) was created, as well as a silhouette score plot (fig. 4). For the elbow plot, there was shown a slight bend at the model with eight clusters, and the silhouette score plot showed a significant peak trend at eight clusters as well. As a result, eight clusters were developed for the model. The PCA package from the sklearn library was leveraged to convert the clusters into a two dimensional structure and visualized (fig. 18).

Table 1

Archetype	Inches	weight	points	rebounds	free throws	three pointers	assists	steals	blocks	value added
0	79.62	219.0	0.39	0.21	1.91	0.71	0.05	0.03	0.03	1.89
1	76.05	194.8	0.22	0.16	0.10	0.07	0.03	0.00	0.00	-0.85
2	80.64	223.4	0.24	0.21	0.65	0.07	0.03	0.02	0.03	-0.51
3	74.92	189.7	0.41	0.12	2.31	1.47	0.09	0.04	0.01	0.75
4	75.34	189.2	0.24	0.14	0.30	0.07	0.05	0.00	0.02	-1.03
5	80.18	220.0	0.26	0.22	0.35	0.01	0.02	0.02	0.08	0.12
6	76.38	196.0	0.26	0.12	0.81	0.74	0.05	0.03	0.01	-1.62
7	77.30	199.06	0.39	0.18	0.33	0.20	0.08	0.08	0.05	0.00

The table above represents the average stat outcomes per minute played across each archetype. Given a profile of cluster characteristics like this, we can give each cluster archetype a descriptive title:

Table 2

Cluster 0:	Game-Centric Forwards
Cluster 1:	GPA Boosters
Cluster 2:	Rotational Bigs
Cluster 3:	High Usage Guards
Cluster 4:	Pass First Guards
Cluster 5:	Imposing Bigs
Cluster 6:	3+D Wings
Cluster 7:	Swiss-Army Knifes

Archetype 0 represents larger players who have a scoring impact teams rely on. The high frequency in which this archetype earns free throws indicates that these players have a high usage rate. On top of that their scoring output is near the top of all archetypes, yielding a name such as Game-Centric Forwards.

Archetype 1 represents a group of small frame players that contribute in little ways across the board. These players are usually walk-ons filling the last few roster spots on a team, while only seeing playing time in extreme situations, such as blowouts. A common term to describe these players is "GPA Boosters" indicating that their lack of talent forces them to take their education more seriously.

Archetype 2 represents the largest overall player in the dataset. Due to their high rebound nature, and low contribution elsewhere, these players are deemed to be "Rotational Bigs" as on many occasions coaches keep a big man who lacks athleticism as depth to plug-and-play for many defensive and rebounding situations.

Archetype 3 are the players you know and love. These players hold the highest point averages, free throw averages, three-point averages, and assist averages indicating they have the ball in their hands most possessions and the offense runs through them. These are "High Usage Guards" referring to their importance within the offense.

Archetype 4 represents players with the smallest frame who contribute in few ways statistically. These players rely on their game sense and passing ability to make up for their undersized frames. Players in this archetype see little minutes and only record assists at an above average rate, giving them the title "Pass First Guards".

Archetype 5 contains players with large frames who record blocks and rebounds at the highest level but fall short for recording stats that take more finesse. These players shine solely on the defensive end and are usually the most athletic big men on the team. Given this profile, these players are labeled "Imposing Bigs".

Players who fall into archetype 6 are larger guards or smaller forwards who take a lot of three-point attempts. These players have a negative VA as they struggle to find other ways to contribute and only score points at a moderate level. Usually, these players are brought in to space the floor and stand in the corner, so the label for this archetype will be "3+D Wings".

Lastly, archetype 7 represents an intermediately sized group of players that offer a balanced skillset to the team. They record steals at the highest rate while also contributing solid points and assists on the offensive end. Given their end-to-end skillset, these players will be labeled "Swiss Army Knifes"

Observing the distribution of archetypes (fig. 5), the largest cluster goes to archetype 6 with over 1400 players assigned that cluster label. These are the most common players to find across CBB rosters due to the replicability nature of this role, player-to-player. These players are mostly tasked with shooting threes and guarding other three-point shooters on the defensive end. The second largest cluster by attendance is the High Usage Guards due to the nature of how a basketball offense has evolved. Teams want to get their hands on guards with a wide range of skills such as shooting threes, playmaking, and getting to the free throw line. This role being the flashiest combined with the increasing popularity of basketball yields many young players working to acquire the skills needed to excel in this position. Archetype 2 and 0 are nearly tied in overall attendance and represent rebounding big-men and scoring focused forwards respectively. These positions are

popular to require for coaches as they ensure physical presence near the rim and non-jump shooting scoring abilities respectively of each archetype.

Classification Model Development

After archetype labeling, the data frame was transformed and compressed into individual game objects. Features in this new data frame included a Home indicator, which labels whether the team was Home/Neutral/Away (1/0/-1), the count of each archetype in the focused team's lineup, and the count of each archetype in the opposing team's lineup. For initial analysis, only the players with the top five highest minutes were counted; these were considered the starters. Lastly, for each group of starters, team and opponent VA was averaged and the difference was calculated as the Net VA Differential, and WP from games prior was averaged and the difference was calculated as the Net Win Probability Differential. This was done to add crucial game contexts such as player skill and pedigree to our predictions. Data was partitioned into an 80/20 train-test split for model training.

The first model developed was a simple neural network model. The TensorFlow library was used and the Keras package was leveraged. What was developed was a feedforward sequential structured neural network with one input layer with 64 neurons and ReLu activation. The second hidden layer was designed with 32 neurons and ReLu activation. The final layer outputs one number and applies sigmoid which ensures that the output number is between 0 and 1. In this network, l2 regularization was set to 0.01 which is an aggressive regularization strategy to severely reduce overfitting but risks significant underfitting as a result. The model was fit using 50 epochs with a batch size of 32, and graded on classification accuracy. This initial model performed a test accuracy of 0.66 and an area under rate of change curve (AUC) of 0.725. The loss curve was visualized for model performance diagnosis (fig. 6).

Another neural network model was developed with a different neural structure and hyperparameter strategy for comparison. The neurons in the first hidden layer were scaled up to 200. This gives the model a larger capacity to detect trends. The second hidden layer takes the 200 learned features and transforms them into another 200. For each hidden layer, ReLu activation was used to detect non-linear trends. A dropout layer was added to randomly turn off 40% of neurons, ensuring the network doesn't rely on just one neuron. Lastly, the output layer returns one value between 0 and 1. L2 regularization was set to 0.002 for a more conservative approach for reducing overfitting comparatively. The model was fit using 100 epochs and a batch size of 5 and graded on accuracy. This model showed a slight improvement of accuracy, exhibiting a test accuracy of 0.67. Model performance was diagnosed using a similar loss curve (fig. 7) which showed a more inconsistent performance nature as the epochs persisted but consistently showed the val curve with less loss than the trained curve, indicating robust performance.

Focused was shifted to leveraging the XGBClassifier algorithm from the xgboost Python library to develop a robust tree model. Extreme Gradient Boosting (XGB) was chosen due to its abilities to handle messy structured data such as our archetype lists as well as its innate ability to explore feature interactive relationships. A simple tree model was initially developed. Initial hyperparameters included 400 estimator trees, max depth 4 to ensure moderately shallow tree structure, learning rate of 0.05 for slow learning, and a subsample of only 80% of data. Results were observed using sklearn's classification report (fig. 8), and compared by performance on both the

training and testing data. The initial tree model performed a train accuracy of 0.72 and test accuracy of 0.66. Discrepancies in accuracy imply slight overfitting and room for model improvement.

A more developed XGB model was produced, with manual hyperparameter adjustment using practices to limit model overfitting. Number of estimator trees were reduced to 150, to prevent heavy training data learning, and max tree depth was reduced to 2 to curb the complexity of each tree model. The resulting performance (fig. 9) showed a train accuracy of 0.69 and a test accuracy of 0.67, reducing the discrepancy between training and testing performance, and matching the test accuracy of this tree model to that of the complex neural network.

Through initial modeling, the features used were observed to be limited to a 0.67 predictive ability. As starters, aren't the only players who provide an impact on the floor during a given basketball game, strategy was shifted to accurately accommodate all players who see minutes in each game. Minute Share[1] was calculated by determining the percentage of the total 200 game minutes everyone played for their team on a given game. An average VA[2] and WP[3] were calculated for each individual player within the dataset to add skill and impact contexts to each individual example of an archetype. As these metrics are global player averages, these metrics represent simplifications and assumptions that a player's value is roughly consistent over time. Since each player can only offer their impact when they are on the floor, player's VA[4] and WP[5] is then weighted by how much they played during a given game. And then the dataset is transformed back into the game-by-game format for modeling, where instead of listing a count of archetypes, it lists how much each archetype had the opportunity to contribute each game, by leveraging the engineering metrics VA and WP[6]. These aggregates only account for the archetype weighted WP values prior to the game's date within the dataset, to avoid data leakage.

Minute Share (1)

$$m_{i,t,g} = \frac{MIN_{i,t,g}}{\sum_{j \in T_{t,g}}^{M} IN_{j,t,g}}$$

Player Average VA (across dataset) (2)

$$\overline{VA_i} = \frac{1}{N_i} \sum_{g=1}^{N_i} VA_{i,g}$$

Player Average WP (across dataset) (3)

$$\overline{WP_i} = \frac{1}{N_i} \sum_{g=1}^{N_i} \Delta WP_{i,g}$$

Weighted VA by Archetype (4)

$$VA_{i,t,g}^{(w)} = m_{i,t,g} \cdot \overline{VA_i}$$

Weighted WP by Archetype (5)

$$WP_{i,t,g}^{(w)} = m_{i,t,g} \cdot \overline{WP}_i$$

Team-Level Aggregation (6)

$$M_{a,t,g} = \sum_{i \in A_a}^{m} m_{i,t,g}$$

$$VA_{a,t,g} = \sum_{i \in A_a}^{V} A_{i,t,g}^{(w)}$$

$$WP_{a,t,g} = \sum_{i \in A_a}^{W} P_{i,t,g}^{(w)}$$

With the dataset optimally enhanced to more accurately reflect real game scenarios, an XGB skeleton was created with a binary objective defined, and an evaluation metric of log loss which penalizes confidently wrong predictions heavily. From this, a hyperparameter tuning algorithm was developed by setting intervals for the hyperparameters and utilizes sklearn's RandomizedSearchCV to take random parameters and split the data into five partitions, train on four and test on the final. This algorithm was run for 50 iterations and was scored on the lowest log loss of the randomized hyperparameters. The resulting optimal hyperparameters were evaluated as:

Colsample by Tree: 0.6002081507981263

Learning Rate: 0.03317316825297632

Minimum Child Weight: 7

Max Tree Depth: 4

Tree Estimators: 297

Reg-Alpha: 0.11040511903162252

Reg-Lambda: 0.0467351899956275

Sub Sample: 0.6888431241882921

Fitting and evaluating the results of the optimal hypermeter tuned model combined with the enhanced model features, we can observe a significant increase in the model performance (fig. 10) as the results show a training

accuracy of 0.78 and a testing accuracy of 0.72, five percentage points higher than the simpler dataset. Feature importance analysis was conducted on this model (fig. 11) showing that the most influential features are the weighted WP values corresponding to archetype 6, 2, 3, and 0. These features directly reflect the most frequent archetype labels (fig. 5) achieved through clustering. The combination of these archetypes combined with their previous game's impact says the most about the predicted game's outcomes

An ensemble model was then developed, combining the predictions from the strongest neural network model, and the hyperparameter tuned XGB. The neural network model was retrained on the new set of features used in the advanced XGB, with the hyperparameters left unchanged. The ensemble voting strategy applied was a soft-voting strategy which is an average of the probabilities across the two models as the model output. This soft-voting strategy was enhanced by running weight tuning to determine how much to adapt to each model's quality. Weight tuning found that the most optimal weight was 0.13 for XGB and 0.87 for NN. This indicates that 87% of the ensemble's outputted probability will come from the NN prediction and 13% from the XGB model's prediction. Training and testing performance were evaluated from this model, where it was reported that the model's training accuracy was 0.745 and the testing accuracy was 0.727.

The CalibrationClassifierCV package from the sklearn library was leveraged and applied to the advanced XGB tree model. Through this process, the package uses cross validation to adjust the output predictions of the model to more accurately represent the real-life winning percentages represented in the dataset. Applying this strategy to our enhanced XGB model, the testing accuracy was improved to 0.73 and the test AUC was improved to 0.81 (both highest performance by a model in our study) (fig. 14).

Results

The model selected for performance analysis was our calibrated XGB model as it yielded the strongest results (fig. 14), and model scalability is optimized with simpler models as opposed to our ensemble classifier with similar performance. Predictive trends were assessed through a series of diagnostic visualizations. Home team win percentage was plotted, faceting for win and loss, with jitter added for optics (fig. 12). As visualized, the sheer density difference between home team wins versus losses is stark and noticeable, indicating that home teams win a majority of CBB games. A perfect classifier would have every plotted point in the loss graph to the left of the 0.5 line and to the right of the 0.5 line for the winning plot. For the winning plot, the majority of points fall to the right side of the 0.5 indicating correct classification. For the losing plot, it is more of a coinflip split where the points are much more spread out, however there are significantly fewer points with high confidence predictions (>90%) in the losing plot, indicating games with high confidence are correctly classified as wins for the most part.

Win rate based on model confidence is then plotted, binning the predictions into intervals of 10% and plotted against the actual win rate of those respective predicted games (fig. 13). A perfectly calibrated predictor would have a real-life win rate as a number within the respective bin. As for the results, bins 40% and above have accurate real-life win rates based on our definition. However, the predicted bins 39% or less have a real-life win rate which exceeds the probabilities within each bin. This shows that the model is underestimating win

probability for home teams that are moderate to significant underdogs, which is important to note when considering model results.

A model calibration diagram was generated to visualize this relationship clearer (fig. 15). This visualization plots the average model probability per 10% bin versus the real-life win rate. As shown, this adds context to the previous binned bar graph, by generating an average within each bin. The model is roughly lined up along the perfectly calibrated guideline, indicating a healthy predictive model.

To demonstrate real-world applicability, we examined how the model would project the 2026 Big Ten tournament, which took place down the road in Chicago. All 18 Big Ten lineups were hardcoded using minute allocation from their final regular season game, on Mar. 6, 2026. The model was run on all possible matchup combinations and resulting win probabilities were displayed in an 18x18 matrix (fig. 16). Each respective win probability was then simulated in a Monte-Carlo style simulative structure in the same format that the real-life bracket reflected. This simulation was iterated 100,000 times, and tournament winning probabilities were represented as the percentage of the simulations each respective team won. What resulted was a probability distribution (fig. 17) showing Michigan as the favorite with a 22.7% winning probability. Purdue was second most favorable with a 17.3% , followed by Michigan State and Nebraska with a 16% chance each. On Mar. 15, 2026, 7-seed Purdue dethroned 1-seed Michigan 80-72 in the championship round of the tournament in a matchup between the two teams with highest projected win probability. In real-life, Vegas sportsbook odds are a great way to evaluate predictive modeling, as it is the most accessible form of data-influenced sports projections. According to sportsbook giant, DraftKings, Michigan had –140 odds (58.33%), while Purdue had 11-1 odds (8.33%) to win it all. In this particular test case, the developed model found heavy overvaluation of the one-seed Michigan, who showed vulnerability prior to the finals, winning by a score of 4 and 3 respectively before ultimately losing to Purdue. Meanwhile, undervaluation was simultaneously exposed with Purdue as their model projected odds were more than double their Vegas projections. It is important to consider this was one demonstrative test case, however, there is optimism that these archetype matchup relationships uncover underlying explanations for game outcomes.

Discussion

Our results show that archetype-based player impact metrics provide important matchup information for strong predictions compared to other archetype model methods. With a data modeling pipeline that was consistently reinforced and improved to a peak of 0.73 classification accuracy supports a hypothesis of cross-team player mixing and matching. 3+D Wings, High Usage Guards, and Game Centric forwards make up the majority of roster construction. Like many other sports, basketball is a copycat industry where coaches observe what works and try and replicate it. Understanding the ways these players interact in a matchup setting on the court in a data driven sense can provide future enhancements, and meta-level breakthroughs where coaches can break the status quo using hidden patterns observed with machine learning. With head-on addressment of current limitations, such as strategically aggregating VA over its current global average state, use of timestamped player substitution data, and human fatigue acknowledgement, this model can be developed into a reliable tool for overall basketball insight.

Due to the scalable nature of archetype labels, coaches can bin incoming freshman into an archetype and project scheme fit for their new players months before they step in the practice facility. Fans can also address hypothetical situations by projecting fantasy matchups by pinning players against each other in a setting that has yet to be seen. To demonstrate this capability, we took the 2025-2026 First Team All-Big Ten team, consisting of Jeremy Fears Jr., Yaxel Lendeborg, Keaton Wagler, Braden Smith, and Pryce Sandfort, and pinned that fantasy squad against the First Team All-SEC lineup consisting of Thomas Haugh, Darius Acuff Jr., Ja'Kobi Gillespie, Tyler Tanner, and Labaron Philon. In this fantasy scenario, the location was set to a neutral court, and all players were assumed to play the entire game (40 minutes). The resulting projection had the Big 10 squad favored with a 64.2% winning percentage, while the SEC only had a 35.8%.

These assessments demonstrate the potential intrigue and usefulness of this model design in situations across the spectrum of realistic to imaginative.

Conclusions

Overall, this study sets to develop a framework for exploring and modeling collegiate basketball results transforming box score data into archetype focused features for matchup analysis. Clustering players into archetypes allowed for crucial minute-weighted predictor development to capture underlying functional compositions of lineups, instead of simple box score counting statistics analysis. Framing the research problem into a role-based representation yielded successful modeling and predictive output.

The calibrated XGB model achieved performed the strongest classification results on the test data with a 73% test accuracy and an AUC of 0.81. Feature analysis further enforced the merit of our approach as archetype-level probability contributions were significant drivers in effective prediction. Suggesting that matchup-based minute distributions of role can be effectively modeled and especially useful for modeling the innate variability of basketball matchups compared to raw team metrics alone.

Through a simulated test case of a real, live event, the model also exhibited practical applicability when identifying inefficiencies in current market evaluations supporting potential value in creating archetype-centric predictions. Limitations remain however, as the model uses globally average player impact metrics in the interest of overall simplicity.

Overall, this work shows evidence that leveraging archetype-based modeling in predictive basketball analysis can offer a scalable and effective approach to forecasting and evaluating team construct, and potential matchups.

Future Work

This model was designed for scalability where the most obvious next step is to include data from farther back in time. Inclusion of all teams and conferences would also be an important follow-up to ensure the model accounts for all teams and playstyles seen in CBB over the most recent years. Number of clusters can be tuned

with respect to final model predictive accuracy, as opposed to silhouette scores. Other model algorithms such as Logistic Regression and Random Forest can be explored and ensembled in attempts to maximize accuracy. Weights can also be explored and applied to the model based on recent team success with intention for a more usable real-time predictive model. An effectiveness curve can also be developed to penalize players who are modeled to play more minutes than they normally would, as factoring in human fatigue is a current blind spot in the model.

Accessibility for data driven tools is of the upmost importance to me. Final datasets will be published on the web. I intend to follow up my work to create an interactive frontend interface which fans can use to mix and match player lineups across different teams, different years, and different game contexts, such as game location.

Appendix

Fig 1

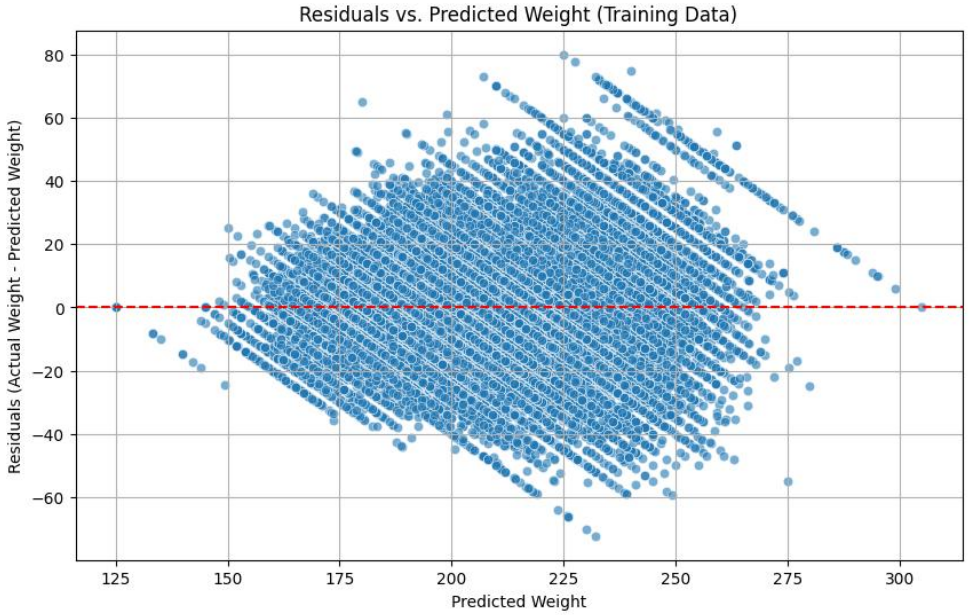


Fig 2

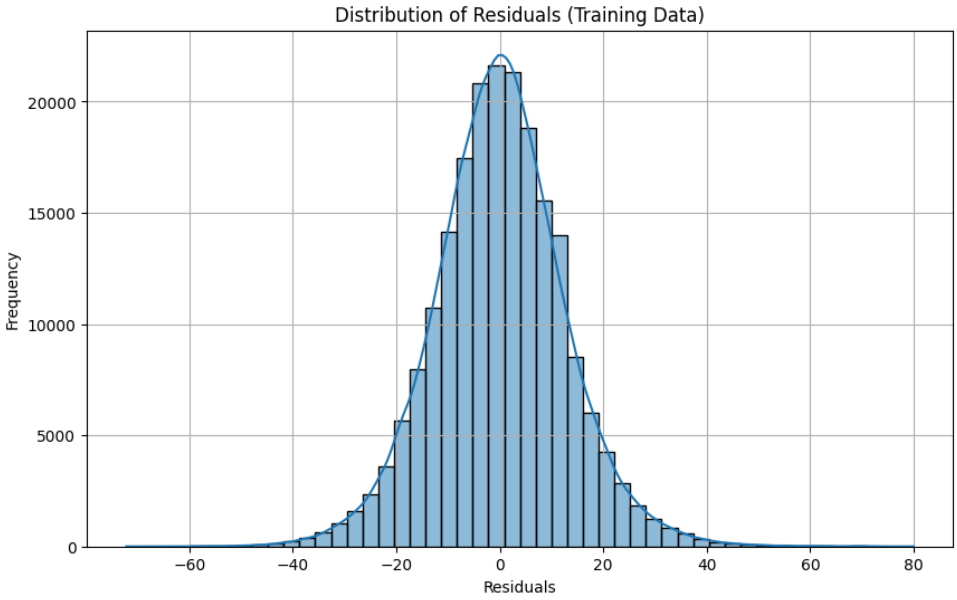

Distribution of Residuals (Training Data)
Frequency
Residuals
0
5000
10000
15000
20000
−60
−40
−20
0
20
40
60
80

Fig 3

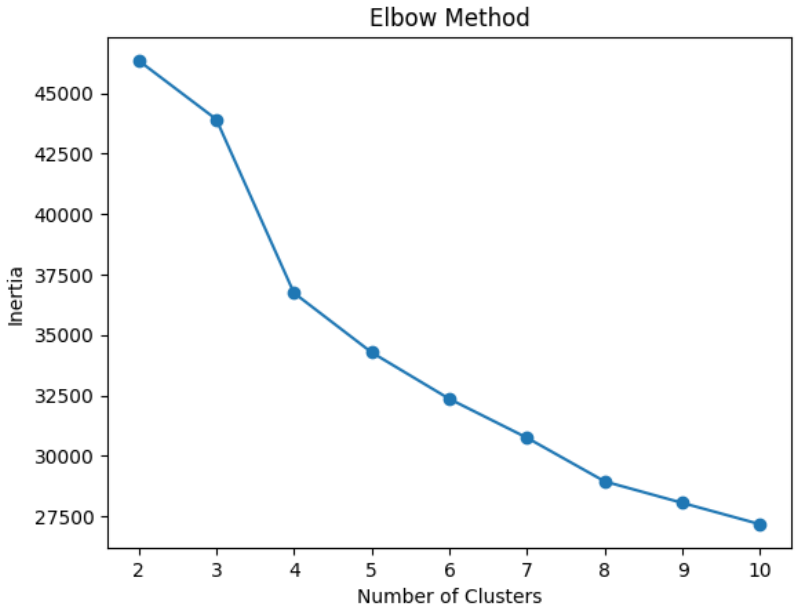

Elbow Method
Inertia
Number of Clusters
45000
42500
40000
37500
35000
32500
30000
27500
2
3
4
5
6
7
8
9
10

Fig 4

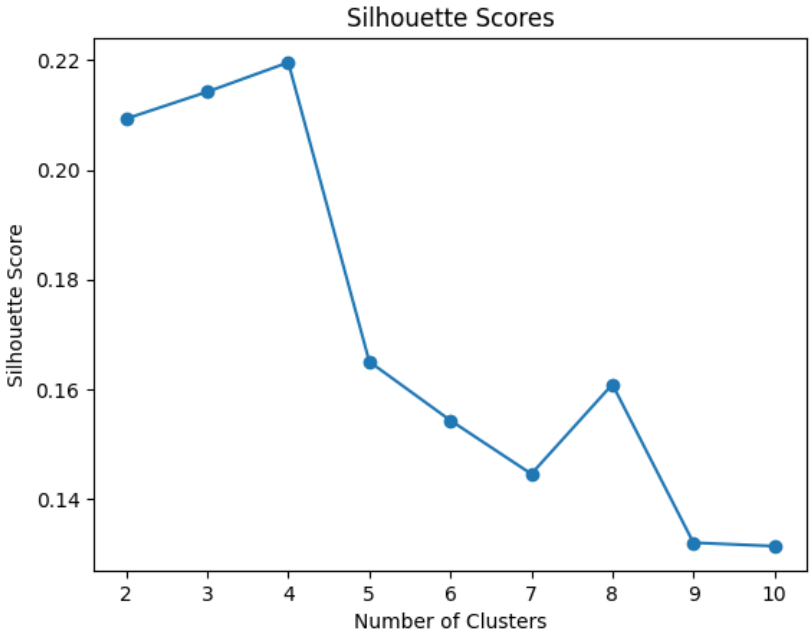

Silhouette Scores
Silhouette Score
0.22
0.20
0.18
0.16
0.14
2
3
4
5
6
7
8
9
10
Number of Clusters

Fig 5

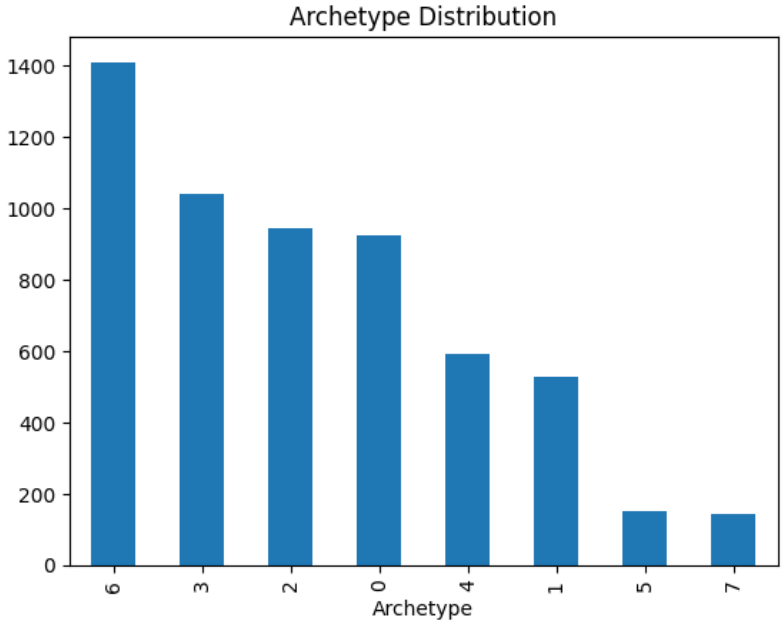

Archetype Distribution
1400
1200
1000
800
600
400
200
0
6
3
2
0
4
1
5
7
Archetype

Fig 6

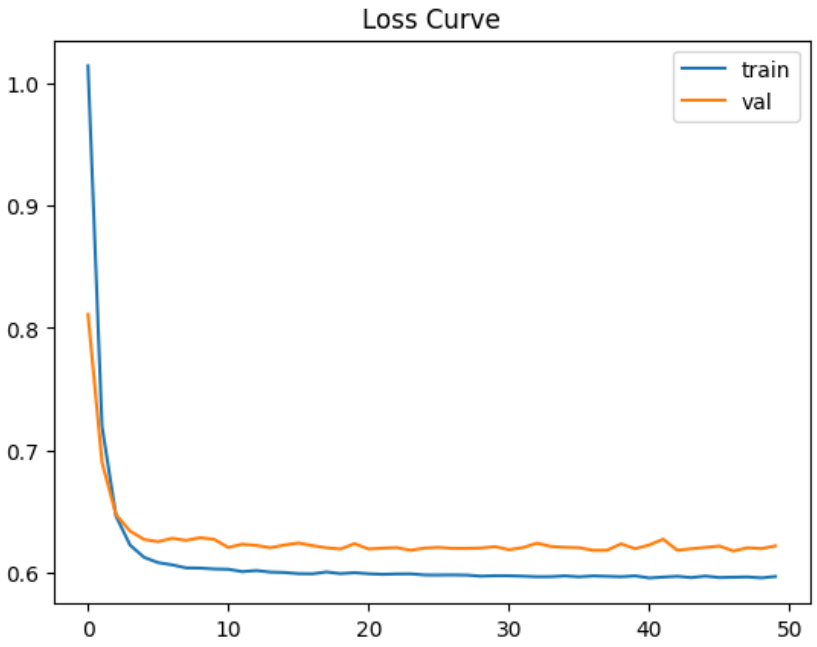

Loss Curve
train
val
1.0
0.9
0.8
0.7
0.6
0
10
20
30
40
50

Fig 7

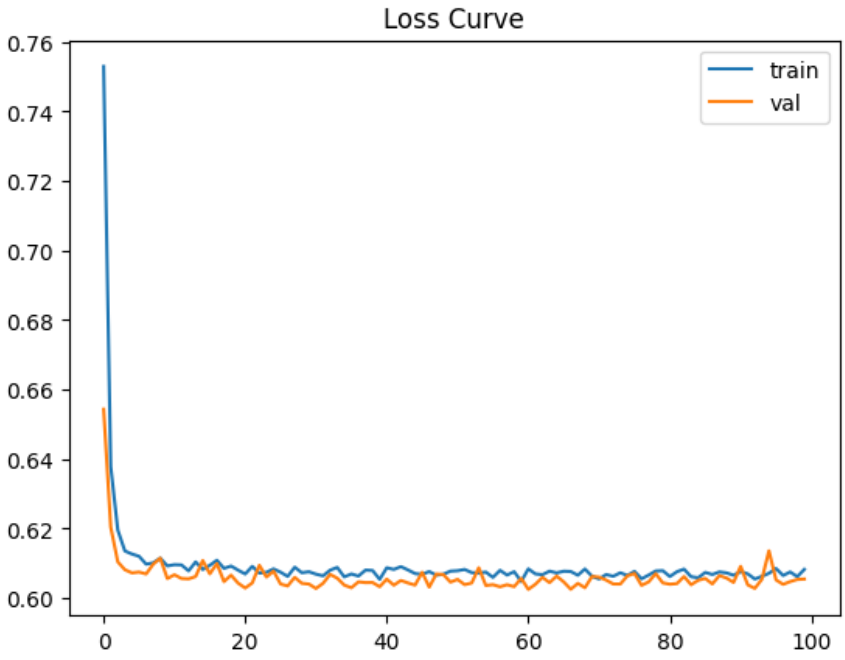
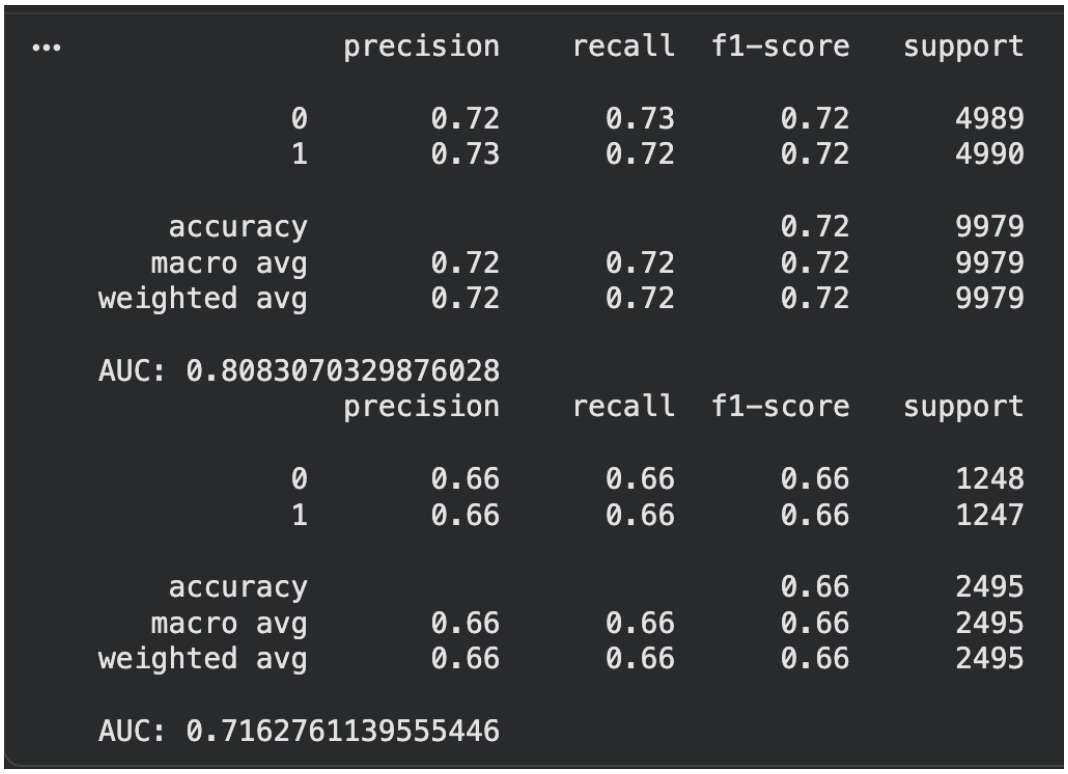
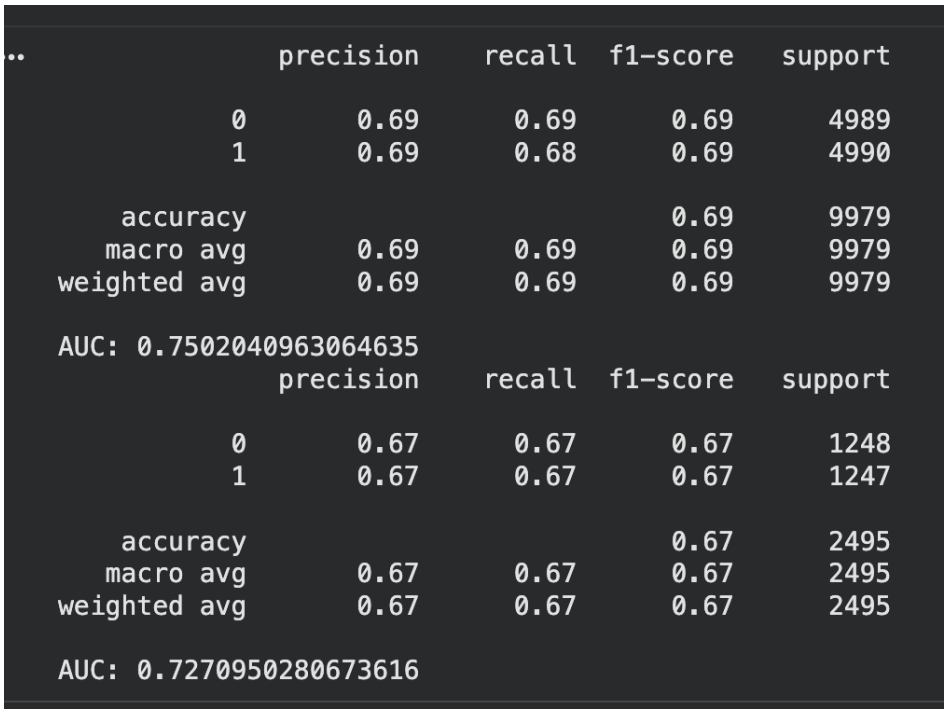

Loss Curve
train
val
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
0.60
0
20
40
60
80
100

Fig 8



```
...               precision    recall  f1-score   support

           0       0.72      0.73      0.72      4989
           1       0.73      0.72      0.72      4990

    accuracy                           0.72      9979
   macro avg       0.72      0.72      0.72      9979
weighted avg       0.72      0.72      0.72      9979

AUC: 0.8083070329876028
              precision    recall  f1-score   support

           0       0.66      0.66      0.66      1248
           1       0.66      0.66      0.66      1247

    accuracy                           0.66      2495
   macro avg       0.66      0.66      0.66      2495
weighted avg       0.66      0.66      0.66      2495

AUC: 0.7162761139555446
```

Fig. 9



```
...               precision    recall  f1-score   support

           0       0.69      0.69      0.69      4989
           1       0.69      0.68      0.69      4990

    accuracy                           0.69      9979
   macro avg       0.69      0.69      0.69      9979
weighted avg       0.69      0.69      0.69      9979

AUC: 0.7502040963064635
              precision    recall  f1-score   support

           0       0.67      0.67      0.67      1248
           1       0.67      0.67      0.67      1247

    accuracy                           0.67      2495
   macro avg       0.67      0.67      0.67      2495
weighted avg       0.67      0.67      0.67      2495

AUC: 0.7270950280673616
```

Fig 10

```
...                precision    recall  f1-score   support

             0       0.78      0.78      0.78      4989
             1       0.78      0.78      0.78      4990

      accuracy                           0.78      9979
     macro avg       0.78      0.78      0.78      9979
  weighted avg       0.78      0.78      0.78      9979

  AUC: 0.8667411391233057
                   precision    recall  f1-score   support

             0       0.73      0.72      0.72      1248
             1       0.72      0.73      0.72      1247

      accuracy                           0.72      2495
     macro avg       0.72      0.72      0.72      2495
  weighted avg       0.72      0.72      0.72      2495

  AUC: 0.8107226574548145
```

Fig 11

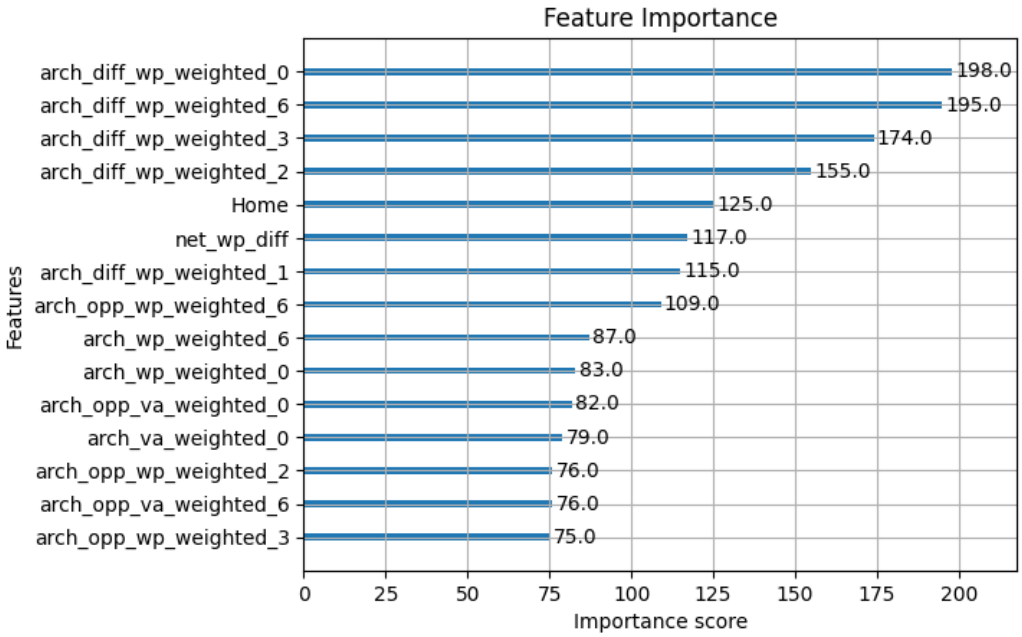


Fig 12

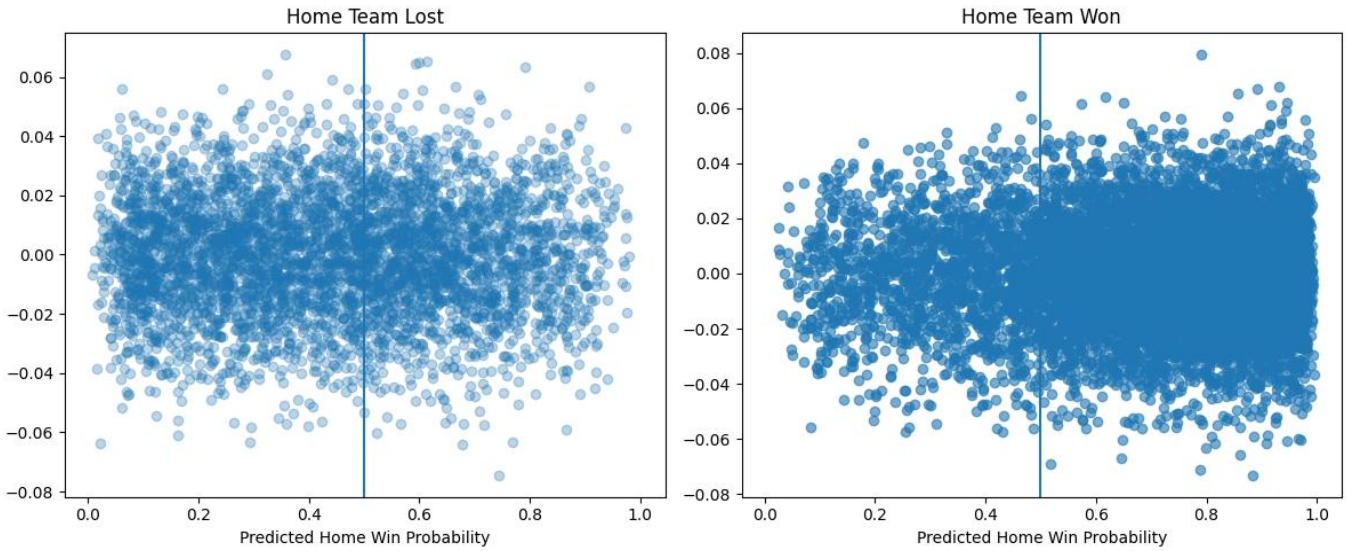

Home Team Lost
Home Team Won
Predicted Home Win Probability
Predicted Home Win Probability

Fig 13

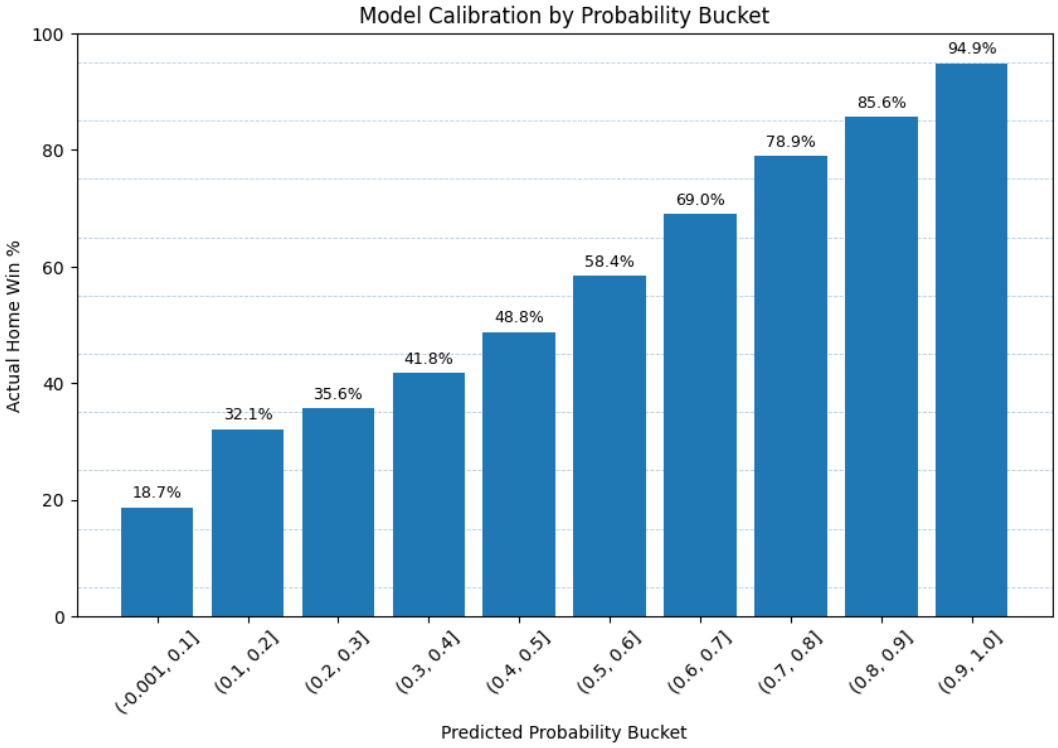

Model Calibration by Probability Bucket
Actual Home Win %
Predicted Probability Bucket
18.7%
32.1%
35.6%
41.8%
48.8%
58.4%
69.0%
78.9%
85.6%
94.9%
(-0.001, 0.1]
(0.1, 0.2]
(0.2, 0.3]
(0.3, 0.4]
(0.4, 0.5]
(0.5, 0.6]
(0.6, 0.7]
(0.7, 0.8]
(0.8, 0.9]
(0.9, 1.0]

Fig 14

```
...               precision    recall  f1-score   support

           0       0.77      0.80      0.79      4989
           1       0.80      0.76      0.78      4990

    accuracy                           0.78      9979
   macro avg       0.78      0.78      0.78      9979
weighted avg       0.78      0.78      0.78      9979

AUC: 0.8674379426321072
              precision    recall  f1-score   support

           0       0.72      0.75      0.73      1248
           1       0.74      0.71      0.73      1247

    accuracy                           0.73      2495
   macro avg       0.73      0.73      0.73      2495
weighted avg       0.73      0.73      0.73      2495

AUC: 0.8118037777846319
```

Fig 15

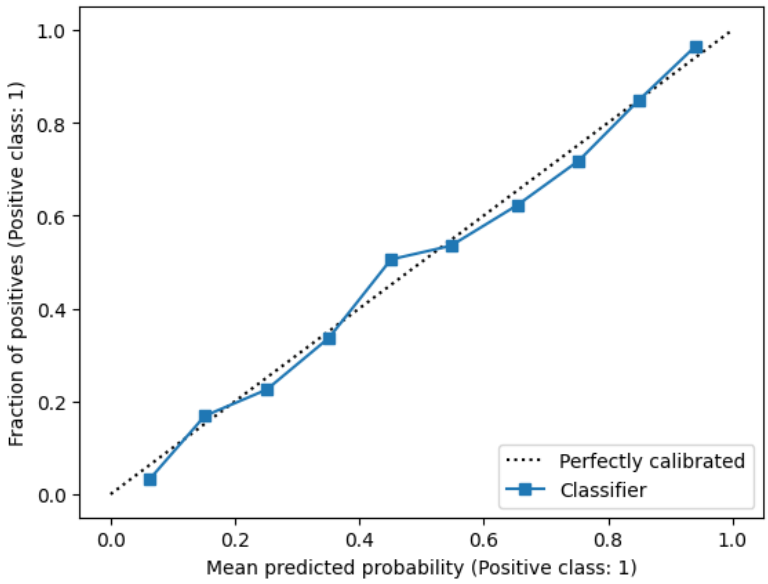


Fig 16

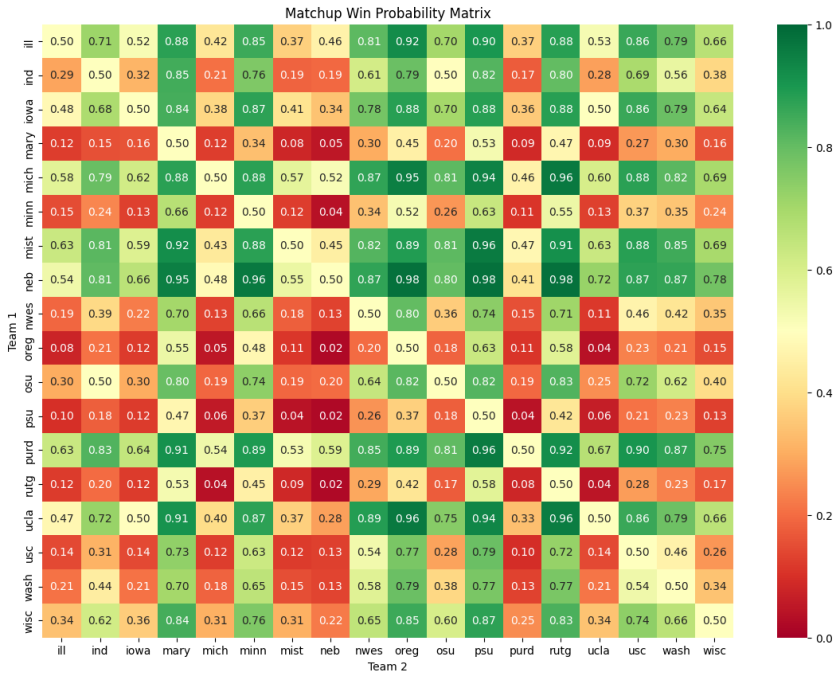


Fig 17

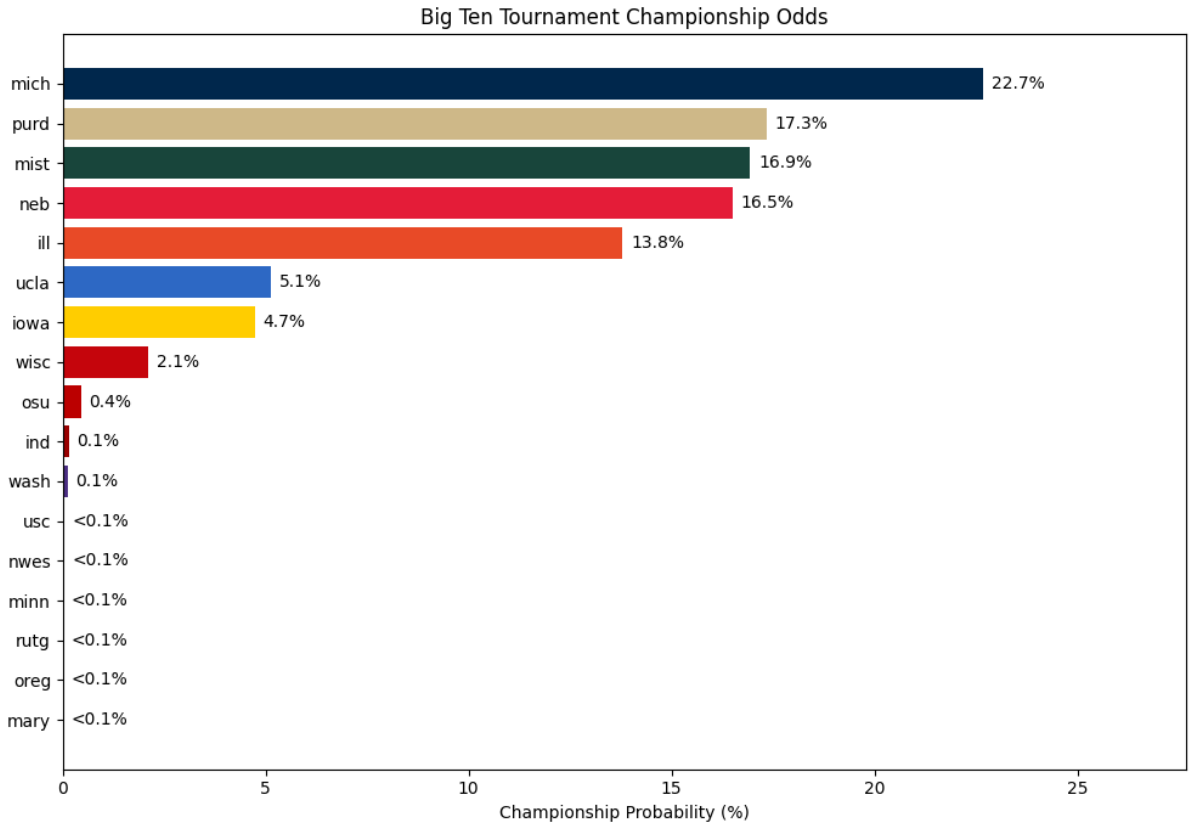
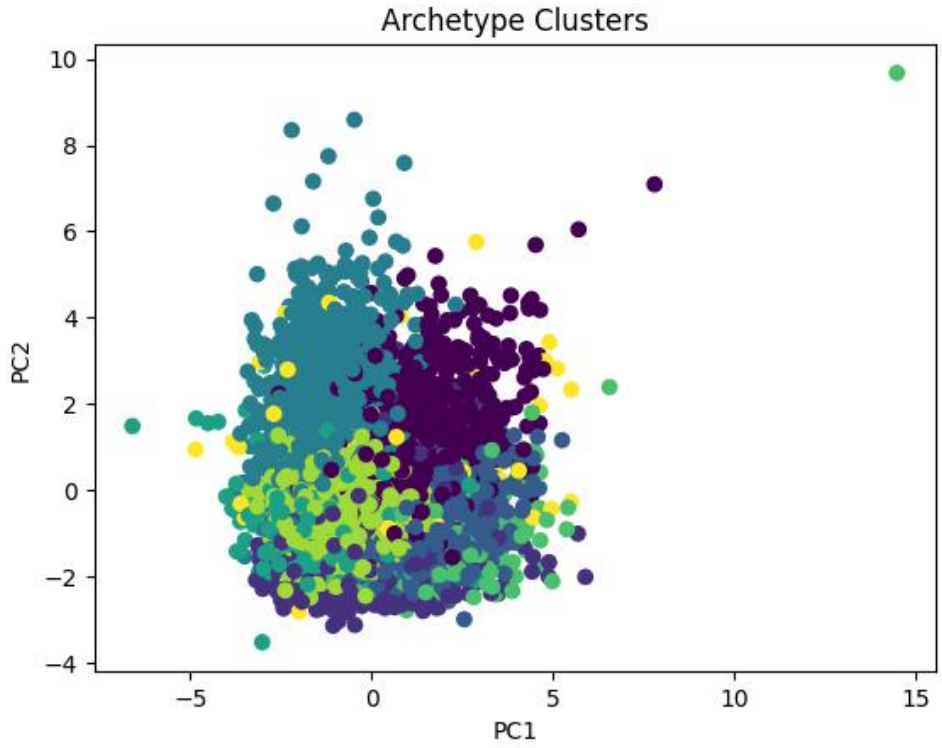


Fig 18



Bibliography

Cohen, Z. (2026, March 9). *2026 Big Ten Conference basketball tournament odds & predictions*. VSiN.

ESPN Reference. (2026, March 15). *Purdue Boilermakers vs. Michigan Wolverines game summary*. https://www.espn.com/mens-college-basketball/game/_/gameId/401851388/purdue-michigan

FuzzyWuzzy Reference. (2020). FuzzyWuzzy documentation. https://github.com/seatgeek/fuzzywuzzy

Google Colaboratory Reference. (2023). Google Colaboratory documentation. https://colab.research.google.com

Matplotlib Reference. (2023). Matplotlib documentation. https://matplotlib.org/stable/index.html

NumPy Reference. (2023). NumPy documentation. https://numpy.org/doc/

pandas Reference. (2023). pandas documentation. https://pandas.pydata.org/docs/

Patton, A. (n.d.). How to make a win probability model. GitHub. https://github.com/anpatton/basic-nbtutorials/blob/main/win_probability/make_win_probability_model.md

Penner, L. S. J. (2025). Player archetypes within basketball: Optimizing roster composition to create a championship team. Frontiers in Sports and Active Living, 7, Article 1639431. https://doi.org/10.3389/fspor.2025.1639431

Salao, C. (2025, April 21). *NCAA transfer portal hits record numbers—again*. Front Office Sports. NCAA transfer portal hits record numbers—again

scikit-learn Reference. (2023). scikit-learn documentation. https://scikit-learn.org/stable/documentation.html

SciPy Reference. (2023). SciPy documentation. https://docs.scipy.org/doc/scipy/

Seaborn Reference. (2023). Seaborn documentation. https://seaborn.pydata.org/

TensorFlow Reference. (2023). TensorFlow documentation. https://www.tensorflow.org/api_docs

XGBoost Reference. (2023). XGBoost documentation. https://xgboost.readthedocs.io/