MGT Madness - Team 15 Progress Report

Group Members: Michael Munson, Matthew Rosenthal, Matthew Royalty, Jeffrey Sumner

Overview of Project:

Project Background:

Every March, millions of Americans come together to watch the NCAA Basketball Championships, AKA "March Madness". The college tournament attracts, which consists of 67 games spread over a 3-week period, has millions of viewers and spectators, generating significant revenue for the NCAA. In 2022, the tournament had 10.7 million TV viewers, 685,000 attendees, and generated an estimated $1.15 billion in revenue [1].

Naturally, March Madness is a gold mine for sports betting. Firms such as DraftKings and FanDuel earn a commission with each bet made, in addition to revenue from their websites and app ads. In order to encourage bettors to their sites and to maximize this revenue, betting companies must have the most accurate models possible. This is a challenging task for even the most seasoned data analysts; games are influenced by a near-infinite number of factors, ranging from "obvious" ones like team statistics to more abstract ones like injuries, momentum, and crowd noise.

The goal of MGT Madness was to create a machine learning model that reliably predicted the outcome of both March Madness and regular season college basketball games at a level equal or better to ESPN's industry-standard prediction models. To achieve this, our main driving question was: What predictors could be used to accurately predict the outcome of a college basketball game?

To answer this question, we collected large amounts past game data, processed the data, selected influential features, trained and validated numerous models, and finally compared these models against each other on various methods.

Overview of Data:

Data Preparation and Cleaning

The project involved gathering data on game outcomes, betting odds, and AP poll rankings for college basketball games between 2012 and 2022. The data was collected using a combination of web scraping and querying APIs.

The data collection process began with using datasets from the `ncaahoopR` library, which included box score, play-by-play, rosters, and schedule files. The resulting data frame served as the basis for amending and creating other features.

Similarly, we used the `hoopR` package to retrieve betting data for each game using the `espn_mbb_betting(<game_id>)` function. Due to the size of the data files, this data had to be stored within three different files.

To collect data on AP poll rankings, the we wrote a web-scraping function that collected data from sports-reference.com using the `rvest` package. The data was then processed using the `tidyverse` package and `lapply()` function to combine scraped data frames into a single table.

Finally, we processed and merged all of the collected data into one data frame using the `data_cleanup.R` script. The script included processing steps such as filtering and renaming variables, geocoding college locations, and merging data frames.

Not all of the data collection went smoothly. There were times when ESPN and other sources throttled or even blocked us from pulling data. This was an unforeseen issue that ultimately required our team to take a step back and rethink our approach. We found a publicly available ESPN API that allowed us to pull the necessary data in a more efficient manner than web scraping.

Once we had our data, we had to determine the right ways to clean our data given our project needs and objectives. We explored many questions, including: Do we want to use statistics based on the last 5 or 10 games? Do we want to focus solely on how a single team performs or do we want to compare how the team performs vs how the opponent performs?

For an in-depth look at how we combined our code and sources, please refer to our Team 15 github.

Data Source Overview:

- ESPN API has a wealth of information ranging from team information (location, logo, colors, jerseys, etc.) to box-score and game-time information
- sports-reference.com contains AP poll ranking data in addition to other ranking sources
- The ncaahoopR data contains information around box scores, play-by-play and team information. The box scores and play-by-play data will be instrumental for our project. This will be one of the main sources for key statistics such as points scored, rebounds, steals, win probabilities, etc. In Table 1 is an example of one of the box scores
  - https://github.com/lbenz730/ncaahoopR
  - https://github.com/lbenz730/ncaahoopR_data
- NCAA Men's Basketball Data:
  - Records from 2000, including game attendance, team records per season, week-by-week Associated Press Poll Records
  - These records are mostly in .pdf format
  - https://www.ncaa.org/sports/2013/11/27/ncaa-men-s-basketball-records-books.aspx

Table 1: Duke Game Sample

player_id	position	MIN	FGM	FGA	3PTM	3PTA	FTM	FTA	OREB	DREB	REB	AST	STL	BLK
4592187	F	30	5	8	0	1	7	10	6	4	10	2	1	
4065653	F	12	0	1	0	1	0	0	0	2	2	0	0	

Data Exploration:

We generated a correlation matrix of our factors and observed several expected relationships. For example, there was a high correlation between a team's defensive rating and the number of blocks/turnovers generated by that same team, as well as between a team's offensive rating at home and the average points per game scored at home. Conversely, we found that there was less correlation between statistics of the home team and those of the away team. For instance, the home team's average points per game was less correlated with the away team's average points per game.

In addition to correlations, we examined the distributions of factors to ensure that they met normality assumptions and could be effectively scaled later analyses. Fortunately, many of our predictors were approximately normal. However there were some predictors that had centers very close to zero and therefore had non-normal distributions. This included game statistics like flagrant and technical fouls.

Initially, we explored modeling with features centered close to zero. However, in some of the modeling attempts, these features would be selected and negatively impact the model's performance. For instance, models trained with features with centers close to zero would sometimes predict only one class of outcome, such as all wins or all losses. To improve performance, we removed such features, resulting in a reduction from 286 to 237 remaining features.

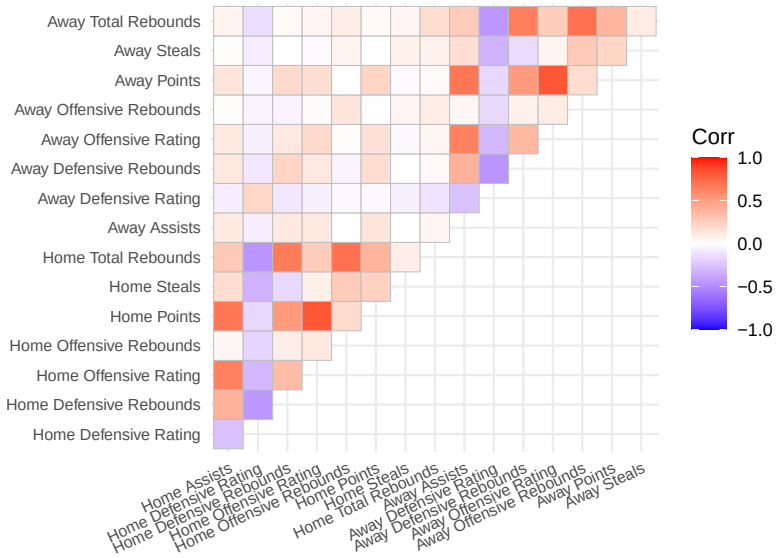

Figure 1: Correlation Exploration

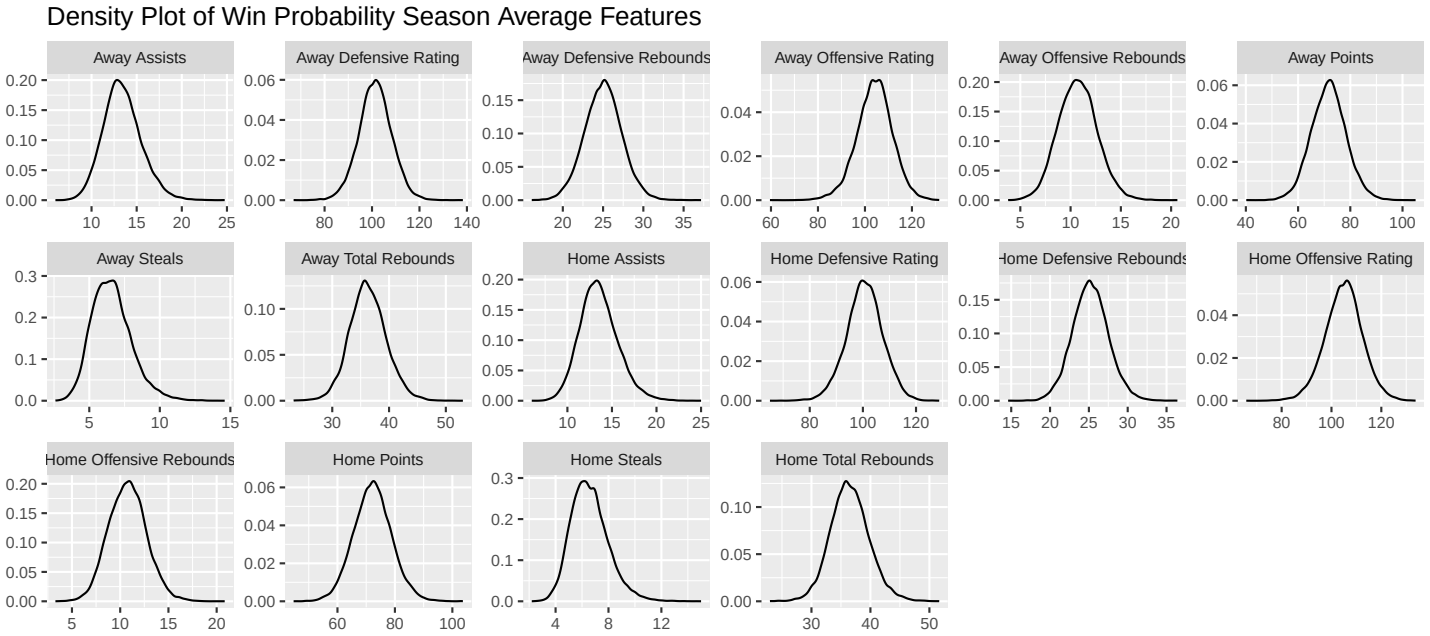

Figure 2: Density Plot Exploration

Feature Engineering

We created predictors for team offensive and defensive efficiency. The efficiency predictors provide useful measures of a team's overall performance by considering both offensive and defensive statistics while not skewing the offensive and defensive metrics based on a team's pace of play. Offensive efficiency is calculated using offensive points per 100 possessions. Defensive efficiency is calculated using defensive points allowed per 100 possessions. Possession data wasn't inherently available in our datasets. We engineered possessions using the following formula:$(FGA - OR) + TO + (Y * FTA)$ where FGA = field goal attempts, OR = offensive rebounds, TO = turnovers, FTA = free throw attempts and Y = 0.44 factor was suggested by kenpom [3].

To provide a more up-to-date representation of team performance we created a rolling 5 game average for each game statistical variable such as points per game, rebounds per game, offensive/defensive efficiency, etc.) for a team over their last five games. By doing so, we aimed to reduce noise in the data compared to a season average, identify trends when a team is performing well, and quickly react to changes such as player injuries or other changes to a team's roster.

In addition to using rolling 5 game averages, we also utilized season average predictors to gain a broader understanding of team performance. To create these predictors, we calculated the average of various team game statistical variables for a team over the entire season. For each of these, we summed the total number of points (or rebounds, assists, etc.) accumulated and dividing it by the number of games played. We hypothesized season average predictors to be useful in providing a more holistic view of a team's performance.

By using both rolling 5 game averages and season average predictors, we aimed to gain a more comprehensive understanding of team performance. Our models were built upon the split of these two techniques, one model for rolling 5 game averages and another for season averages.

The last feature we created was distance traveled to the game site (includes neutral game sites). To create this feature, we collected data on game site longitude and latitude, home arena information and away arena information. The distance calculated was the straight-line ellipsoid distance through the `distVincentyEllipsoid()` function from the `geosphere` package.

After our data collection, cleaning and feature engineering, we had a dataset with 45,518 data points; each representing a single game. Each data point contained 237 features which contained game statistics (rebounds, field goal attempts, etc.) and our engineered features.

Overview of Modeling

As stated previously, the goal was to model the outcome, of any particular NCAA basketball game. Our response variable was whether or not the home team wins or loses the game. We explored common techniques used to predict win probabilities such as logistic regression, probit regression and decision tree classification. In order to do this, we split our modeling out into multiple parts:

1. Splitting the data
2. Baseline models
3. Feature selection
4. "Simple" models with feature selection
5. "Enhanced" models with feature selection
6. All model comparison

The baseline, simple and enhanced modeling procedures were how we progressed a single type of model into its more generalized form. These three procedures were also deployed upon other model types and datasets. For instance, we: explored logit modeling using the rolling 5 game average dataset (as opposed to the season average

dataset), trained probit models on both season average and rolling 5 game average datasets. We also created extreme gradient boosted decision tree models. Finally, we created an ensemble model from all our models. The overall results of each modeling procedure are in Table 3.

Splitting Data

To ensure the validity of our models and to help avoid the risk of overfitting, we randomly split our data into train, test, and validation sets with 60% going into the training set, 20% into the test set and 20% into the validation set. The validation set provided us a way to rank our model's accuracy and aid in model selection. The test set provided an evaluation of our final model. Below is a summary of each dataset with the number of observations, total wins and winning percentage to understand the distribution of wins across each.

Table 2: Train, test and validation bias comparison

type	Wins	Total Games	Winning Percentage
Test	5731	9104	0.6295035
Train	17272	27310	0.6324423
Validation	5746	9104	0.6311511

The split above does appear to be even as far as winning percentage goes and therefore no varying bias across the different types that could throw off our modeling. However, we do see that we have a clear bias towards wins as essentially each dataset has a 63% `Winning Percentage`. This may lead us to over-predict wins across all types.

Baseline Modeling

With our data split into train, test and validation, we predicted outcomes using all features. This exercise is meant to give us something to compare to and hopefully beat as we attempt to improve out models. For the purposes of this paper, we will focus on the comparing the test unless specified otherwise.

ESPN Baseline Let's first examine the ESPN baseline. ESPN predicts outcomes before and during most sporting events. Below is the contingency table and associated metrics:

Based on these results, the ESPN baseline accuracy is 72.8%.

Team 15 Baseline To establish a performance baseline for our own win probability predictive models, we created a simple logistic regression model using all parameters in our dataset for both rolling 5 game averages and season averages. We anticipated a decent model albeit very overfit. Below were the results for the season averaged logit model. The results for the rolling 5 game averages dataset were comparable and will be examined further in our paper:

Based on the results, our baseline accuracy built upon season averages is 71.8%. Overall, this is a decent model but is likely to be overfit and unstable long-term.

From the summary of the baseline models, about 40 features (of the original 78) were statistically significant. Of note, the coefficient of the factor of home_distance / away_distance from our logit model initially confirmed the common assumption that there is a home field advantage. These results may change as we continue to develop our models.

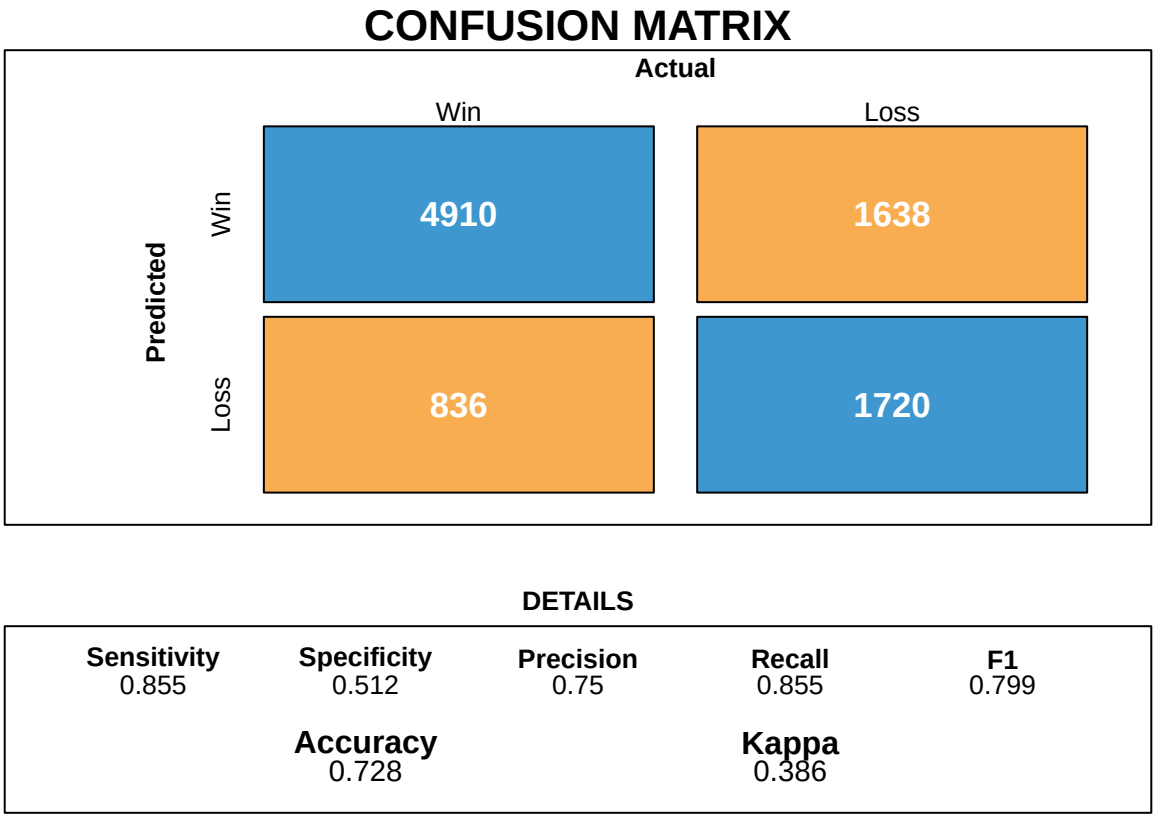

Figure 3: ESPN Baseline Comparison

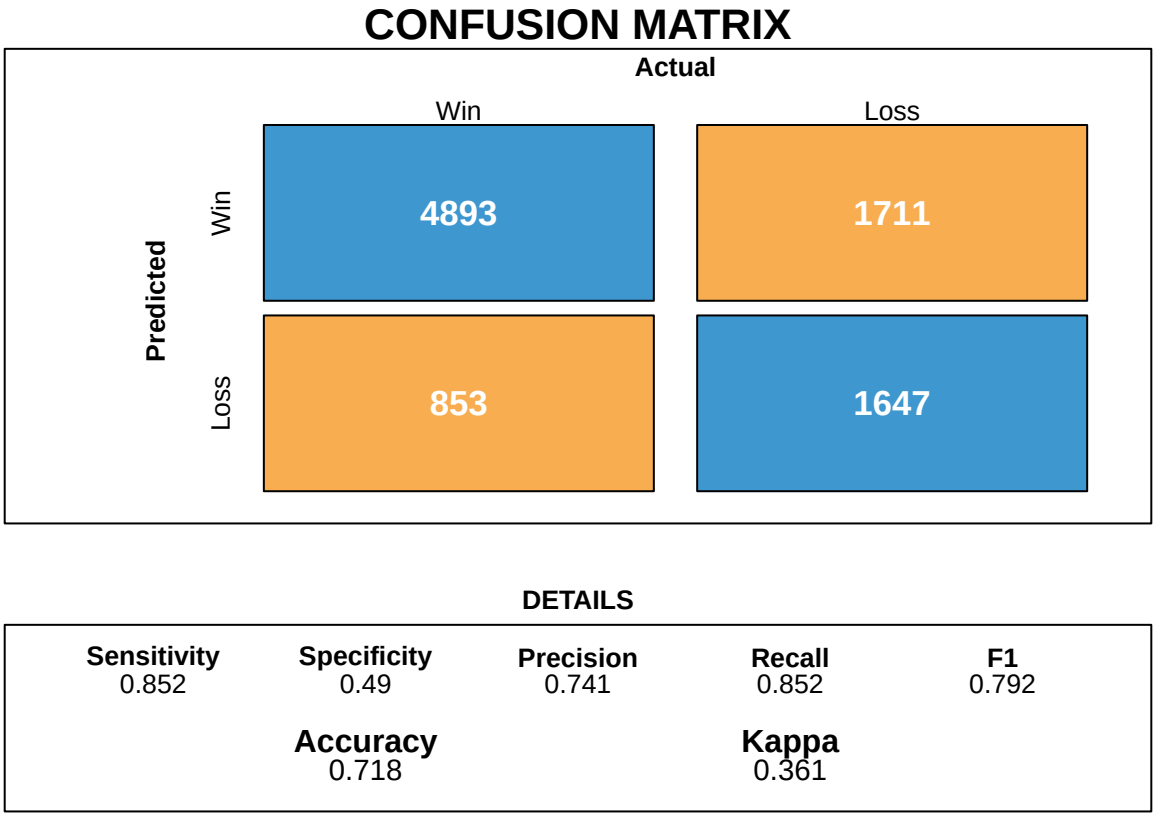

Figure 4: Baseline Season Average Logit Model Confusion Matrix

Feature Selection Exploration

In order to explore feature selection, we looked at stepwise regression using the `MASS` package as well as LASSO regression using `glmnet`. For stepwise, we tested bidirectional elimination. For both feature selection techniques, AIC was used as the model improvement criterion. Both, stepwise and LASSO, produced similar results. Ultimately, LASSO regression was selected due to overall performance due to how quickly it returned results versus stepwise regression. Running stepwise regression using large number of predictors was not as efficient as LASSO regression due to how stepwise regression interactively adds and removes predictors based on metrics until a subset of predictors is found. LASSO regression provided us with a subset of predictors without sacrificing performance training our models.

The training set was used to fit our lasso model which was created to select our optimal set of predictors. Using the `cv.glmnet` function, we performed 10-fold cross-validation on our lasso model.

To find the best lambda value for LASSO, we used `cv.glmnet(family = "binomial", alpha = 1, nfolds = 10)`. We then used the coefficients using the lambda value that was one standard error away from that minimized the cross-validation error to select our significant predictors from an original list of ~78 potential predictors.

Altering the coefficient cutoff changed the selected predictors. We changed the cutoffs multiple times until we found a good balance between number of predictors remaining and logistic regression model performance. Ultimately, we decided to limit our predictors to ones with coefficient not equal to zero.

The results of our LASSO exploration upon the season averaged data led to 30 features being selected (compared to 37 in the baseline). The results of our LASSO exploration upon the rolling 5 game average data led to 44 features being selected (compared to 40 in the baseline). All of the features selected in the season averaged LASSO were included in the rolling 5 game average LASSO. Both included predictors such as: offensive ratings, blocks, field goals attempted, assists, turnovers, and steals. Of note is ratings and distance traveled are still chosen as significant predictors for both LASSO feature selections.

Simple Models

After predictors were selected with LASSO regression, we trained a logistic model upon both the rolling 5 game average and season average datasets. These so-called "simple models" were trained with the features selected by LASSO regression and no other processing of the data.

Relative to the baseline logit model, the simple logit model had very similar metrics. This was nice to see considering that the simple logit model was trained on less than half of the features as the baseline model. Therefore, increasing the generality of the model without significantly sacrificing performance.

Enhanced Models

For the enhanced set of models, our goal was to further protect against overfitting and increase the generalization of our models. The training data was normalized and mapped to principal components via `tidymodels::step_pca()`, which were automatically picked by to include at least 80% of the explained variance. We used `glmnet` versions of the logit, meaning that the lambda was chosen automatically to be the most optimal one. It used 10-fold cross validation through `rsample::vfold_cv()` and optimized on ROC and AUC.

The enhanced model resulted in a very small decrease to our accuracy to specificity and an increase to sensitivity upon the validation dataset. This showed that the enhanced processing was able to maintain the same levels of performance as the baseline and simple logit models while minimizing multicollinearity via principal component analysis.

Model Comparison Matrix

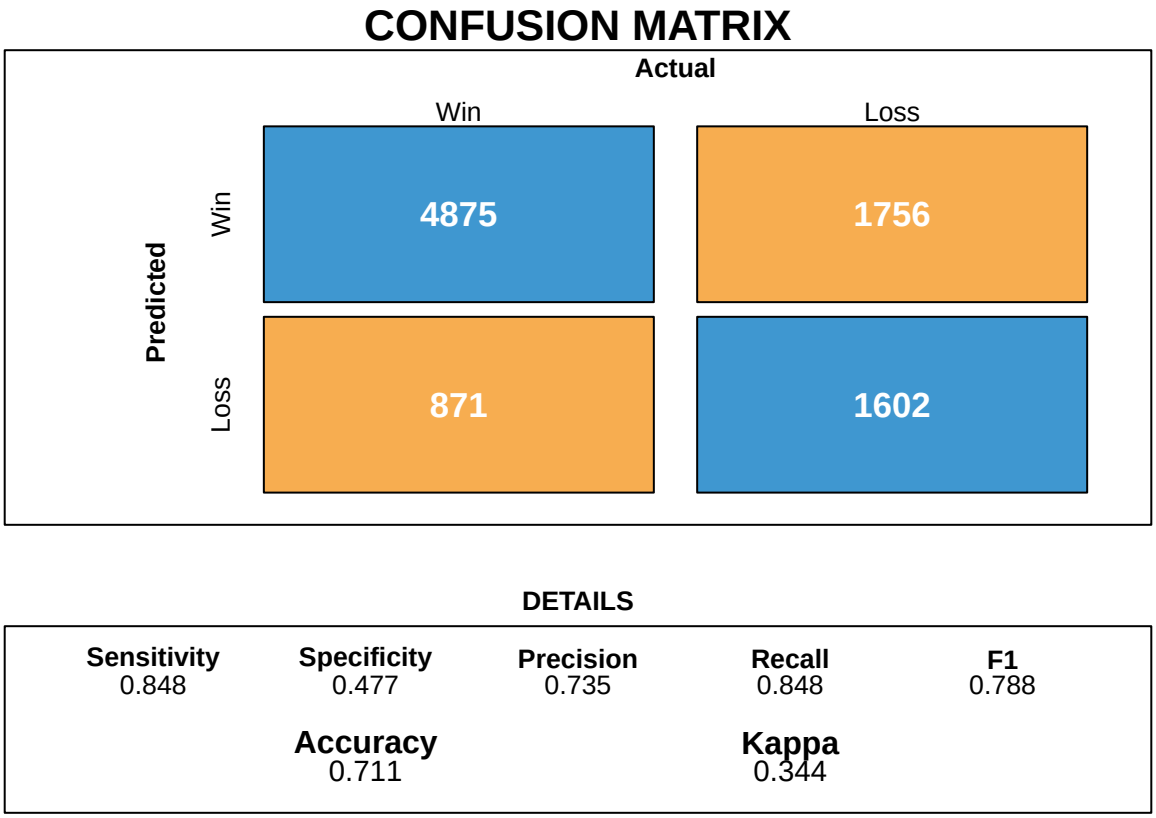

Figure 5: Simple Season Average Logit Model Confusion Matrix

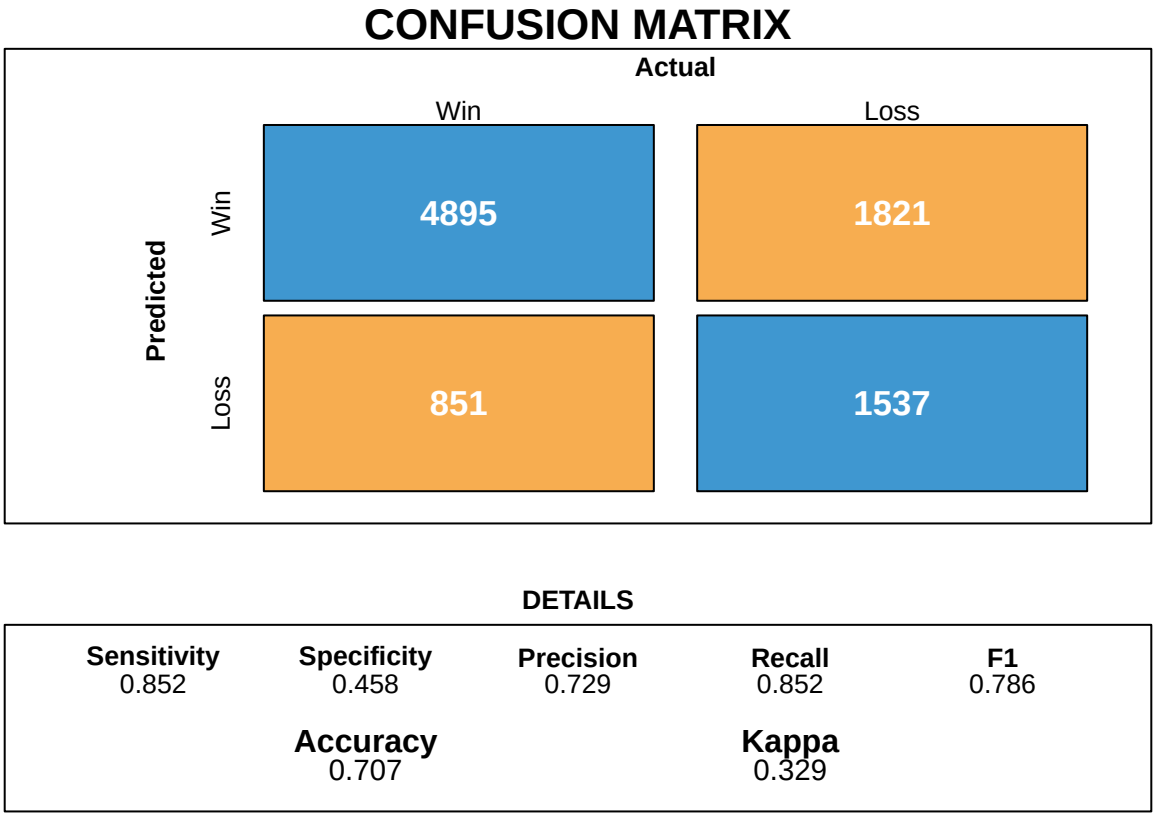

Figure 6: Enhanced Season Average Logit Model Confusion Matrix

Table 3: All Model Comparison matrix - Test Data

Model	Specificity	Sensitivity	Accuracy	Precision
ESPN Pre-Game	49.36%	85.29%	72.00%	74.11%
Baseline Season Logit	46.13%	83.77%	69.80%	72.54%
Simple Season Logit	45.36%	84.23%	69.80%	72.37%
Ensemble Model - Season	44.86%	84.61%	69.90%	72.28%
Simple Season Probit	45.03%	84.28%	69.70%	72.26%
Enhanced Season Logit	44.62%	84.80%	69.90%	72.24%
Enhanced Season Xgb	43.64%	84.16%	69.10%	71.73%
Ensemble Model - All	40.88%	86.98%	69.90%	71.43%
Baseline Rolling Logit	40.05%	84.94%	68.30%	70.65%
Simple Rolling Logit	39.58%	85.24%	68.30%	70.56%
Simple Rolling Probit	39.40%	85.40%	68.40%	70.54%
Ensemble Model - Rolling	38.48%	85.88%	68.30%	70.34%
Enhanced Rolling Logit	36.97%	85.92%	67.80%	69.84%
Enhanced Rolling Xgb	37.12%	85.60%	67.60%	69.82%

Throughout the analyses, we separated the rolling 5 game average data from the season average data. Given metrics in the above table, we saw that there were minor differences between the performance in accuracy and precision of the same type of model on the different training datasets. When we focus on our business case of sports betting, these minor differences could lead to large profit differences. ESPN did a good job maximizing precision and accuracy compared to each of our models. The maximized precision leads them to predict less false positives and in return would stop bettors from betting on losing games.

Rolling 5 game average vs Season average

Based on our results, each model trained on season average data outperformed the models trained on the rolling 5 game average data in terms of specificity, accuracy and precision. The higher precision from our season average models made them better suited for betting purposes as it limited our false positives.

Logit vs Probit

We were also curious if there were any discrepancies between the logit and probit models. From the results, you can see that the simple probit model performed extremely closely with the simple logit model, usually within 0.5% of all performance metrics. This is not an unexpected result as the two models are of similar architectures. If a choice was required between logit and probit in regards to betting, logit tends to maximize precision relative to probit. The overall difference is negligible in low-stakes betting~~, which is something a broke grad student would be participating in~~.

Decision Trees

Decision trees represented our exploration into using a different type of model (since logit and probit are extremely similar). Our boosted tree models were created using the `parsnip::boost_tree()` function. The hyperparameters that needed tuning were: depth of each tree, minimum number of observations in each terminal node, the required reduction in loss for a split to occur, the proportion of the training set to use for each tree, the number of variables to consider for each split, and the learning rate. To find the optimal hyperparameters, a latin hypercube grid search was used and the number of trees was held constant at 100.

Decision Trees vs Logit and Probit Our decision tree models did not have a baseline so we compared them to logit and probit. Overall, the performances of the tree models did not differ significantly when trained on either the season average data or the rolling 5 game average data. The largest difference being in specificity of the models while varying the type of averaging of the data. The tree based model was the lowest ranked model between both the rolling 5 game average and season average models. In terms of betting, the logit and probit models were more reliable as the precision metrics were an improvement over the tree model.

Ensemble

An ensemble model is the average of multiple models with the goal to combine the predictive power of each into a single model. We created three total ensemble models for season average, rolling 5 game average and one for everything. Our accuracy and precision did not change significantly but we believe this model was an improvement and provided more stable out-of-sample predictions than each individual model by themselves.

Modeling Challenges

During our enhanced modeling exploration, many of the models trained on the rolling 5 game averages data would only predict a certain class (such as predicting all wins or all losses,). We believed this was because the rolling 5 game averages features selected from LASSO were sparse and not well correlated with game outcomes. For instance, some of the predictors selected were "flagrant fouls," of which there were mostly zeros in the data.

In addition, we had a large training dataset that led to longer runtimes for models like lasso and boosted decision trees. In a real-world setting we would weigh the benefit of using models like those compared to their compute cost.

Overall Results

In general, our modeling efforts were able to come close to ESPN's historical predictions and could be used as an alternative to ESPN's predictions in order to inform bettors. Given ESPN's market strategy of being the number one source in sports experiences and knowledge, we'd say that our modeling efforts proved to be comperable. One advantage our models had over the ESPN model is that we knew exactly what created the prediction as opposed to ESPN's being more of a proprietary black box.

While accuracy and precision were similar between our season average and rolling 5 game average models, we considered the season models to be better suited for betting given their increased precision and higher specificity.

Where to explore next:

Betting is all about risk-tolerance. Our models predicted probabilities of a win and each win was determined from a threshold of 0.5. If we wanted to be more risky bettors we would decrease the threshold from 0.5 in order to predict more winning games to bet on. If we felt that the least risky route was best, we would increase our threshold from 0.5 in order to have a higher confidence in our win predictions.

Our feature selection was done using LASSO regression but we briefly explored stepwise feature selection. Additional attempts could use the stepwise approach to minimize AIC and to find a different set of predictors for our models.

There were many features that we were interested in incorporating, such as time between games, individual player statistics and injuries data. Although distance traveled had a significant effect, we also hypothesized that we would be able to see an effect due to direction traveled (i.e. if a player traveled to another time zone and lost an hour of sleep).

We planned to perform a parallel study on predicting game attendance in addition to win probabilities. However, upon generating the content for game outcomes, we realized that it would not be feasible to represent both analyses

within this group project. We were interested in using projected attendance to build some sort of advertising CPM estimates for March Madness.

Finally, we would look to operationalize our models and simulate real-life betting scenarios using our predictions to determine how they work in practice. In addition to betting, these models could be used to build a March Madness bracket since they are generalized to fit any NCAA Men's Basketball game.

Works Cited

- [1] Bubel, Jennifer. "How Much Money Do Universities Get for Going to the NCAA March Madness Tournament?" *Diario AS*, 28 Feb. 2023, https://en.as.com/ncaa/how-much-money-do-universities-get-for-going-to-the-ncaa-march-madness-tournament-n/.
- [2] Parker, Tim. "How Much Does the NCAA Make off March Madness?" Edited by Jefreda R Brown, *Investopedia*, Investopedia, 9 Mar. 2023, https://www.investopedia.com/articles/investing/031516/how-much-does-ncaa-make-march-madness.asp#:~:text=In%202022%2C%2045%20million%20Americans,see%20the%20heftiest%20cash%2Dout.
- [3] Pomeroy, Ken. "The Possession", *Kenpom*, 19 Mar. 2004, https://kenpom.com/blog/the-possession/#:\~:text=T
- [4] Korpar, Lora. "March Madness Betting Expected to Exceed $3 Billion, Set All-Time High", *Newsweek*, 14 Mar. 2022, https://www.newsweek.com/march-madness-betting-expected-exceed-3-billion-set-all-time-high-1687917
- [5] Coleman, Madeline. "March Madness: How a fan used a machine to nail his bracket", *Sports Illustrated*, 31 Mar. 2021, https://www.si.com/college/2021/03/31/march-madness-fan-trained-machine-predict-bracket-will-geoghegan
- [6] Consoli, John. "Advertisers Go Mad For March Madness", *TV News Check,* 16 Mar 2022, https://tvnewscheck.com/business/article/advertisers-go-mad-for-march-madness/