NBA Game Predictions

DS 5110 Spring 2022

Christian Prosser, Peter Causey, Trent Cork (gpm6tf, hss3nd, txc5ve)

Abstract

The potential outcome of a basketball game can change quite quickly with a key shot, or a pivotal turnover. How much a play changes the outcome of the game is dependent on several factors, such as how much time is left, the current score differential, who gets the ball after the play, is a team shooting consistently better, etc. In the modern NBA era, there is detailed reporting for each play. Using Basketball Reference's play by play game data, via Kaggle, our goal was to create a model for predicting win percentage in an NBA game after each play. This could help teams understand to what degree of risky (or conservative) measures a team should take in efforts to win the game. A more practical use of this too would be giving the average fan a glimpse into how likely their team is going to win.

It should be noted that game win probability prediction is not something we pioneered- this is a widespread practice. However, our intentions were to utilize PySpark and its pipeline feature to find the most optimal model for game prediction from 'big data', which in this case was the raw play by play data of NBA games. We first transformed the data into usable features for win prediction. After that, we created functions for each model type that would prepare the data in a pipeline, build the model, and tune the hyperparameters with cross validation. Results would report back to a shared file each time a model type was run.

The champion model was a Support Vector Machine Model with the features score differential, score differential times the inverse of seconds left, and which team had possession. The model had an area under the ROC of .8274 meaning that is considerably better than random guessing. However, though models with other features did not perform quite as well, by creating models based on these features we are able to provide meaningful insights on the effects of statistics a team can control. For example, a model on assist, turnover, block, foul, and rebound differential can show a team how much more or less likely they are to win for every statistic they have more or less than the other team. We have also noted future work in the conclusions section as we believe there is opportunity to further progress insights on game win probability from what we have found so far, including analyzing how prediction accuracy changes throughout the game.

Data and Methods

Data Preparation

The data that was used for this project was Basketball Reference's play by play data from the 2015-2016 season to January 20th of the 2020-2021 season. By nature of the data, the response variable is already balanced- every game a team has to win or lose. This data did not require much cleaning, however, feature creation was implemented to provide data that was more useful and geared towards predicting game probability. The original data was more of raw reporting of plays, the features created provided more information on team's performance and game status in a numerical format that could be used for model training. Each play is one row in the data.

Initial Feature Creation

Below is a list of the original features of the data:

URL	AwayTeam	ShotOutcome	ReboundType	LeaveGame
GameType	AwayPlay	ShotDist	ViolationPlayer	TurnoverPlayer
Location	AwayScore	Assister	ViolationType	TurnoverType
Date	HomeTeam	Blocker	TimeoutTeam	TurnoverCause
Time	HomePlay	FoulType	FreeThrowShooter	TurnoverCauser
WinningTeam	HomeScore	Fouler	FreeThrowOutcome	JumpBallAwayPlayer
Quarter	Shooter	Fouled	FreeThrowNum	JumpBallHomePlayer
SecLeft	ShotType	Rebounder	EnterGame	JumpballPoss

Table 1

For data on each play, the original data recorded the current score, a string describing the play in "HomePlay" or "AwayPlay" and a string in the associated type of play with a description and player. This did not directly give statistics for each team in a game. With that, a feature, 'Team' was created to represent which play a team was for. This was created by assigning the play (row) to a team based on whether HomePlay of AwayPlay was populated.

Our goal was to predict the win probability for each team in a game, with that we aimed to create a prediction model that predicted each team's win probability separately. To do that, we

created a row for each team in a play. Otherwise, we would be predicting whether a team won or lost, but the model would only be based on a binary prediction of one of the teams- not both teams. We created a function to create a row for each team in the game, we referred to this transformed data as ‘stacked’ data.

Creating the ‘Team’ feature would allow for other statistics to be tracked for each team, including:

- HasPossession
    - Creates a binary variable for which team (Home or Away) has possession
    - Utilizes Shooter, Assister, Fouled, Rebounder, ViolationPlayer, FreeThrowShooter, TurnOverPlayer, JumpballPoss. Based on if these features are populated by a team.

- Teams’ cumulative statistics
    - Assists, blocks, turnovers, rebounds, fouls, shooting percentage, and free throw shooting percentage.
    - This was created using Window.PartitionBy.
    - From this data features were created to represent the delta between teams on these statistics.

In addition to these features that were created for performance statistics of each team, game status features were also created:

- Won - Response variable
    - The original feature would include the team’s name, this allowed us to have a binary variable for if the team in the respective row won or not.
    - This is the binary response variable that models were created to predict.

- ScoreDiff
    - Score difference between the teams using HomeScore and AwayScore, relative to the team in the respective row.

- SecLeftTotal
    - Utilizes Quarter and SecLeft to determine total seconds left in the game. The original SecLeft feature was the seconds left within the quarter.
    - This accounted for overtime games as well.

The script for cleaning and transforming the features included writing the data to a shared folder in Rivanna.

Exploratory Data Analysis

Of the 40 columns in the original data set, the features of interest after preparing the data could be distilled to:

- Game Status:
    - ScoreDiff
    - SecLeftTotal
    - HasPossession
- Team Stats- both for each team individually as well as the deltas.
    - Shots
    - Assists
    - Blocks
    - Turnovers
    - Fouls
    - Rebounds
    - Free Throws

Other variables, such as time of game, location, and date were not considered. It should also be noted the team was not considered as a categorical variable. We aimed to create a model that was independent of a team's past performance.

Below is a chart showing the distribution of team wins. In the basketball community it is widely thought that some teams are better in the 'clutch', especially if they have more experience. Considering a team's record in close games is an area to explore for future work.

Figure 1

Correlational analysis in Figure 2 (below) shows moderately positive correlations between ScoreDiff, AssistDiff, and Rebound Diff with Won. Essentially teams with more assists, rebounds, and points than their opponents at any point during the game may be more likely to win - correlation is not causation but our model will prove this out later. Turnovers have a strong negative correlation with shots on goal implying that teams that have more turnovers tend to have fewer shots on goal. A fascinating insight here is that teams that have more free throws than their opponent tend to have fewer shots on goal and fewer assists compared to their opponent.

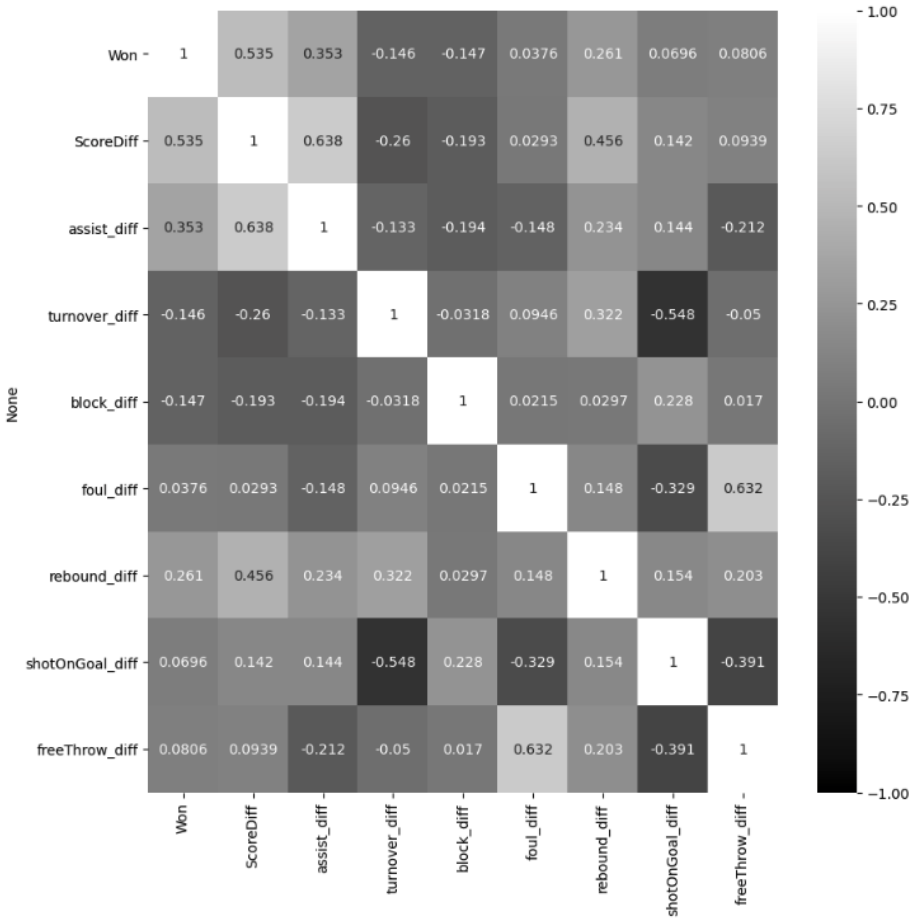

Figure 2

We ran preliminary logistic regression models to better understand what features had larger weights for probability prediction. ScoreDiff was much higher than SecLeft, which based on subject matter knowledge, we knew could not be correct- both mattered. Also, this initial logistic regression model was not giving probabilities with magnitudes much different than 50%, even at the end of the game. What we realized was that SecLeft alone could not predict whether a team won or lost. A game could be over but without the score the probability of a team winning or losing would still be 50/50. With that, we created a feature that was a function of both time and score. Time (Seconds left) needed to be an inverse though to match ScoreDiff of having a positive correlation with win probability.

This led to the creation of SecLeftTotalInverseTimeScoreDiff. We drew inspiration from an article in the Journal of Sports Analytics titled “Evaluating NBA end-of-game decision making” by Patrick McFarlane, who is a quantitative analyst for the Philadelphia Phillies where a feature for a function of time and score differential was presented (McFarlane 2019). However, our feature, SecLeftTotalInverseTimesScoreDiff is a simplified version of this. The intent was to have a feature that was more easily interpreted.

Model Construction

Functions were created to construct and test each model. Below is an overview of the steps of the function:

- The function would first build a pipeline to prepare the data for model construction, utilizing the features the user calls for. This allowed us to test models with various features.
- Models were built and tuned for the best hyperparameters using Cross Validator.
    - One season of data was used for cross validation - it was further split 70/30 for training the cross validator and evaluating the cross validator. The hyper parameters chosen by the cross validator model was then used to train the same type of model on a second season of data to validate our results.
    - Evaluation metrics for selecting hyperparameters was Area Under the Receiver Operating Characteristic (AUC-ROC) curve. The ROC curve plots the true and false positive rate at different thresholds for classification based on predicted probability. The larger the area under the ROC represents a model that generally has better prediction performance across the different thresholds.
- After hyperparameters were tuned via cross validation models, the model was applied to the test data (30% split) and prediction results were collected, which included:
    - AUC-ROC
    - The Area Under the Precision Recall Curve (AUC-PR)
        - The PR curve plots the precision - the rate at which positives are predicted correctly, and recall - likelihood to detect true positives. The PR curve tells how well a model can perform on predicting accurately and overall detection.
    - True Positives
    - True Negatives
    - False Positives
    - False Negatives
    - Note: Threshold for prediction is 50%.
- Results were saved into a shared CSV file on Rivanna which kept track of the model type used, features used, run time, and evaluation metrics.

Model learning methods tested:

Model	**HyperParameters**
Support Vector Machine	Iterations, Aggregation Depth

Logistic Regression	Iterations, Regularization Parameter
Random Forest	Bins, Depth, Number of Trees

Table 2

The script allows the user to select the number of cores they want to run and filter the season data fed into the model to only a portion of each game if desired.

Results

Through the model functions we created mentioned above, we were able to test several model types with several different features, each with hyperparameters tuned. These results are shown in Table 3.

Overall, the champion model that provided the highest performance for game win prediction was the Support Vector Machine with ScoreDiff, SecLeftTotalInverseTimesScoreDiff as the features as it had the highest area under ROC, this is shown in row 34 of Table 3. This model had a precision rate of 73.6% and recall of 73.2%.

Our results for several different features in the pursuit of the best model also provided unintended insights that can be utilized. Although team statistics were not a part of the champion model, models created on features that teams can control- team performance statistics, give insight into how these affect win probability. For example, with a model based on the assist, block, turnover, rebound, and foul battle throughout the game, there is still an area under the ROC curve of .7362 as seen in row 1 of Table 3. This model's results demonstrate that a model without ScoreDiff and only team statistics can perform almost as well during regulation. This is insightful because we have created a model that coaches can use to strategy with their team, as these statistics are somewhat controllable goals, and the benefits of working towards them can clearly be seen. Furthermore, based on the coefficients, we can tell how the change in each of these individual statistics affects a teams' win probability. However, this does not apply in overtime, the model performs worse, as shown in row 9 of Table 3. We believe this is because if a game makes it to overtime, teams are performing very similarly.

We also looked at models that were only trained and tested on data from the final quarters, as the script we developed allowed us to filter seconds remaining in the game. Initial model performance results showed a large improvement as seen in rows 42 and 43 of Table 3.

	model_type	list_features	special_description	cv_best_hyperparameters	val_area under_roc	val_area under_pr	Precision	Recall	Accuracy	F1 Score
1	Logistic	['assist_diff'; 'turnover_diff'; 'block_diff'; 'foul_diff'; 'r		{'maxIter': 20; 'regParam': 0.1}	0.7362	0.7289	66.9%	66.8%	66.9%	66.9%
2	Logistic	['assist_diff'; 'turnover_diff'; 'block_diff'; 'foul_diff'; 'r	First Quarter On	{'maxIter': 10; 'regParam': 0.1}	0.6231	0.6063	58.7%	59.3%	58.8%	59.0%
3	Logistic	['assist_diff'; 'turnover_diff'; 'block_diff'; 'foul_diff'; 'r	Quarter = 1	{'maxIter': 20; 'regParam': 0.1}	0.6289	0.6132	59.1%	59.7%	59.2%	59.4%
9	Logistic	['assist_diff'; 'turnover_diff'; 'block_diff'; 'foul_diff'; 'r	Quarter >= 5	{'maxIter': 20; 'regParam': 0.1}	0.6257	0.5962	59.4%	59.4%	59.4%	59.4%
31	Logistic	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff'; 'HasP		{'maxIter': 20; 'regParam': 0.1}	0.8196	0.8218	73.6%	73.2%	73.5%	73.4%
32	Random Forest	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff'; 'HasP		{'maxBins': 3; 'maxDepth': 3; 'numTrees': 10	0.7949	0.7708	68.1%	84.5%	72.5%	75.4%
33	Random Forest	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff'; 'HasP		{'maxBins': 3; 'maxDepth': 3; 'numTrees': 10	0.7949	0.7708	68.1%	84.5%	72.5%	75.4%
34	SVM	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff'; 'HasP		{'aggregationDepth': 5; 'maxIter': 50}	0.8274	0.8340	73.6%	73.2%	73.5%	73.4%
38	Logistic	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff']	First Quarter On	{'maxIter': 20; 'regParam': 0.5}	0.6444	0.6367	61.3%	55.2%	60.2%	58.1%
42	Logistic	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff']	Quarter >= 3	{'maxIter': 20; 'regParam': 0.5}	0.9011	0.9014	83.0%	80.1%	81.8%	81.5%
43	Logistic	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff']	Quarter >= 4	{'maxIter': 10; 'regParam': 0.5}	0.9419	0.9434	87.6%	84.4%	86.2%	86.0%
44	Logistic	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff']	Quarter >= 5	{'maxIter': 10; 'regParam': 0.1}	0.8753	0.8808	82.1%	69.0%	77.0%	75.0%
45	Random Forest	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff']		{'maxBins': 3; 'maxDepth': 3; 'numTrees': 10	0.7907	0.7680	67.0%	86.5%	72.0%	75.5%
52	SVM	['ScoreDiff'; 'SecLeftTotalInverseTimesScoreDiff']	Quarter >= 4	{'aggregationDepth': 5; 'maxIter': 20}	0.9477	0.9498	87.6%	84.4%	86.2%	86.0%
55	Logistic	['ScoreDiff']		{'maxIter': 10; 'regParam': 0.1}	0.8179	0.8198	74.7%	70.9%	73.4%	72.7%
56	Random Forest	['ScoreDiff']		{'maxBins': 3; 'maxDepth': 3; 'numTrees': 10	0.7793	0.7551	66.6%	86.4%	71.5%	75.2%
57	SVM	['ScoreDiff']		{'aggregationDepth': 3; 'maxIter': 10}	0.8179	0.8198	72.3%	76.0%	73.4%	74.1%
59	Logistic	['SecLeftTotalInverseTimesScoreDiff']	First Quarter On	{'maxIter': 20; 'regParam': 0.1}	0.6447	0.6373	61.3%	55.2%	60.2%	58.1%
66	Logistic	['turnover_diff']		{'maxIter': 10; 'regParam': 0.1}	0.5487	0.5438	53.9%	45.8%	53.3%	49.5%
67	SVM	['turnover_diff']		{'aggregationDepth': 3; 'maxIter': 10}	0.5487	0.5438	52.9%	60.8%	53.3%	56.6%

Table 3

Conclusions

Through the script we built that included a pipeline to prepare data, we were able to tune and test several models. Testing several models allowed us to understand how prediction can work with various features, which was important for this dataset as different users will get more value for contemplating certain features than others, such as coaches wanting to tell their teams to minimize turnovers, or bettors just wanting the best model.

Considering that the model was evaluated on predictions throughout the game, we believe a precision rate of 73.6% and recall rate of 73.2% with the champion model is effective enough to be used by teams in their contemplation of risk vs. conservative strategy and for fans to understand if their team will win the game, as it is considerably better than random guessing, but there is room for improvement, users may want to see better rates than this before devising game strategy or betting money on it. As mentioned, a majority of the current models have focused on the entire game. We expect accuracy would increase in the final minutes of the game, when strategic decisions could matter more, as seen in preliminary results mentioned in the results section. For future work we recommend further analyzing how prediction accuracy changes throughout the game, and conducting further model tuning and selection based on training and testing in the final minutes.

The models specific to team statistics arguably provide more meaningful insights, as teams can see the benefit of trying to achieve certain deltas with their opponents. Even if these prediction performances aren't as high, they are higher than random guessing and worth pursuing to give a team the best advantage they can get.

References

McFarlane, Patrick. “Evaluating NBA End-of-Game Decision-Making.” *Journal of Sports Analytics*, vol. 5, 2019, pp. 17–22., https://doi.org/10.3233.