6.867: Final Project

1. Introduction

The National Basketball Association (NBA) is currently one of the most popular sports in the world. Just a few months ago, over 30.8 million viewers tuned in to watch Game 7 of the NBA finals between the Cleveland Cavaliers and the Golden State Warriors making it the third most viewed event in the U.S in 2016 (just behind the Super Bowl and the Academy Awards). As a result of this popularity, the betting market for the sport is enormous. NBA commissioner Adam Silver predicted the market to be around $ 400 billion.

This report is concerned with exploring a variety of supervised learning techniques in hopes of being able to make predictions of game outcomes that are more accurate than the predictions made by NBA experts who set the betting line for each game. Before diving too deep into the algorithms, models, and predictions, we want to provide a quick overview of the structure of an NBA season as well as the various types of bets you can make in the NBA.

First, it is important to understand that the NBA is comprised of 30 teams split into two conferences (East and West). These 30 teams will each play 82 games across the regular season meaning that 1230 NBA games are played in total each season. For each game of the season, there are multiple bets that can be made. Below, we will discuss the two most common bets:

a) Win/Loss: This is the most basic bet that can be made. The gambler simply picks which team he believes will win the game and if he is correct then he will win money.

b) The Spread: The spread is a bit more advanced than simply win/loss. When a gambler bets against the spread, he either bets that the favorite shall win by more than the spread or that the underdog will lose by less than the spread. Typically, the spread is expressed as a negative number, which signifies the expected margin of victory for the favorite. In order to place a bet against the spread, a gambler generally needs to bet $110 for the chance to win a $100 payoff. The amount required for the chance to win $100 is generally expressed in parentheses.
e.g. Golden State Warriors -7.4 (-110)
Cleveland Cavaliers +7.4 (-110)

points scored
points allowed
Elo score
Games played
Win rate

Table 1. Basic statistics

The gambling authority generally selects the spread with the goal of splitting the betting money down the line in order to essentially make arbitrage with 0 risk. Due to the fact that the spreads are created in this manner and not with accuracy in mind, we believe that there is room for improvement in terms of predictive performance.

2. Data collection

For this project, we needed to gather a great deal of historical NBA game statistics in order to train our models. To collect season data, we primarily used publicly available data from `http://basketballvalue.com/downloads.php`.
These datasets contained time series data in the form of matchup logs for all NBA games ranging from 2005 to 2012. Matchup logs basically describe the interactions that occur between 5-person units. A new entry is created every time a substitution is made in a game. By running a python script on these matchup logs, we were able to derive and construct a variety of statistics that would later serve as features. For seasons 2005-2008, the matchup logs were a bit simpler, so we were only able to derive basic statistics including points allowed and points scored (Table 1). The matchup logs became a bit more advanced in seasons 2008-2011, which allowed us to derive more sophisticated statistics including offensive and defensive rebounds (Table 2). Note that while we had data for the 2011-2012 season we ultimately decided not to use the data because a lockout occurred during this season which shortened the season to 66 games.

# defensive rebounds	# offensive rebounds
# possessions	Home court advantage
# home points scored	# home points allowed
Home court games played	Home court win rate

Table 2. Advanced statistics

For all seasons, we also constructed a new feature called the Elo score, which is a reflection of a team's performance by taking into account strength of schedule. What this means is that if two teams possess identical records, one of the teams may actually have a higher Elo score if they had to face stronger competition in the past. In our literature review, this was not a feature that had been incorporated before in machine learning so we were particularly excited about its implications.

Many of the features generated for training and testing our predictor were derived from this game data. In order to create a general proxy for how strong a team was, we calculated running averages of points allowed, points scored, win rate, Elo, etc. and used these as input features to our classifier. These running averages were initialized to equal numbers for all teams at the beginning of the season and were subsequently updated as more data became available. Because these numbers were arbitrarily initialized to equal at the beginning of the season, however, running averages near the beginning of the season serve as less reliable data points, as outliers can significantly affect the running average.

These statistics were therefore plotted over time throughout the season in order to approximate when the running averages for each of these statistics stabilized to reliable averages. An example plot is given in Figure 1, in which the running average win rate for a given team is plotted over course of the season.

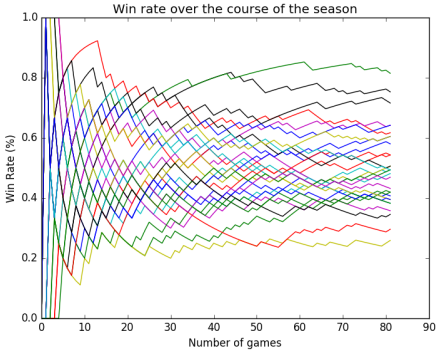

Figure 1. Win rate over time

In general, the running averages of the game data stabilized after approximately each team's twentieth game of the season (around the 1/4th mark of the season). When generating our training, validation, and test sets therefore, we only used data after each team's twentieth game in order to ensure that the features extracted from the data were representative of true team performance.

In order to compare our algorithm and our predicted spreads against Vegas and other betting authorities, we created a scraper to scrape relevant websites with both current and historical spreads. We primarily scraped this information from `http://www.sportsbookreview.com/betting-odds/nba-basketball/`.

3. Normalization

The need for feature normalization arises because our features are on different scales. For example, the average points a team scores every game is around 100 whereas the average number of offensive rebounds a team gets per game is around 15. In order to normalize the features we took a Z score approach that centered all the data at 0 and set the standard deviation to be 1.

4. Linear classifier

As a baseline, we used our own implementation of a linear classifier on basic features including running averages of home/away team points scored, points allowed, Elo score, games won, games played, and win rate. We ran this classification in order to predict which team wins/loses in a given matchup as well as in order to predict spreads on these matchups.

4.1. Win/loss prediction

Based on the 2006-2007 season, in predicting win/loss, the linear classifier had an accuracy of 64.3% on training data and 63.3% on testing data. After adding further features, such as home court advantage, were added to the model, the linear classifier has an accuracy of 72.3% on training data and 66.9% on testing data based on the 2008-2009 season.

4.2. Spread prediction

We additionally explored the ability of the classifier to predict spreads on games, in which the margin of victory is taken into account instead of a simple win/loss. In this case, the accuracy metric was the absolute value of the difference between the predicted spread and the

	Win/Lose Accuracy	Spread Error
Basic features	63.3%	10.07
Full features	66.9%	13.60

Table 3. Classifier performance on test dataset

M=1	**M=2**	**M=3**	**M=4**	**M=5**
10.15	11.35	11.013	12.40	11.32
M=6	**M=7**	**M=8**	**M=9**	**M=10**
12.05	33.26	33.56	17.11	150.23

Table 4. Average spread error for M values

actual spread. Using basic features on the 2006-2007 season, the average absolute value spread error was 9.31 points for training data and 10.07 points for testing data.

After adding further features, the average absolute value spread error decreased to 8.53 points for training data but increased to 13.60 points for testing data. This would imply that the classifier might be overfitting to the features that are provided as input to the algorithm, which would be expected for more complex architectures but was surprising in the context of the linear classifier explored here.

These results for the performance of the linear classifier are summarized in Table 3.

4.3. Polynomial basis regression

Similar results were found when we used a polynomial basis and allowed the degree of the polynomial fitting the data to grow. Out of values for $M \in [1, 10]$, we found that a regression model with $M = 1$ performed best, where $M = 10$ on the other side of the spectrum led to extreme overfitting. The average spread error is summarized in Table 4 for each of the different possible values for M, and graphs for representative values of M are illustrated in Figure 2. Note that for ease of illustration of data here, only the differential win rate is used as the feature vector when plotting.

As the data naturally suggested a fairly linear trend and because $M = 1$ performed the best during polynomial regression, moving forwards we primarily considered linear regression and improvements to the linear classifier in order to improve overall performance of the classifier.

4.4. Multiple seasons of data

We additionally explored the use of generating training, validation, and testing datasets across multiple

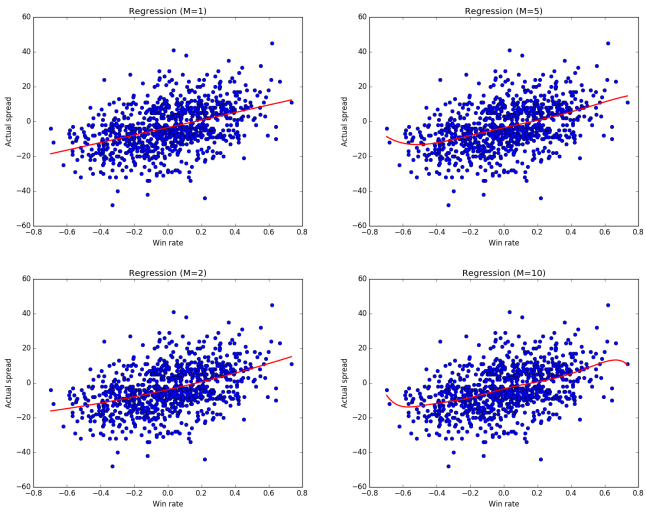

Figure 2. Win rate vs. actual spread

seasons of data. In this case, because we train the model against certain features of the teams playing (e.g. running average win rate, average points scored, average points allowed, etc.), and not the skill level associated with the team itself, we can aggregate data across multiple seasons. This holds true even if teams and their respective players have changed significantly between seasons, because the features we draw from the data are running averages calculated from the season so far.

When we run the linear classifier on three seasons worth of data, in which the training, validation, and testing datasets are each made up of a season of data (about 1300 games), the accuracy of the algorithm improves as expected. More specifically, when run on three seasons of data, the linear classifier is able to classify win/loss of matchups with an accuracy of 72.3% on training data and 68.6% on testing data. In terms of spreads, the linear classifier predicts spreads with an average spread error of 8.98 points on training data and 9.29 points on testing data. Thus, increasing the size of the data set has a fairly significant improvement in the accuracy of the linear classifier when compared with the 66.9% accuracy previously obtained for win/loss predictions and 13.60 spread error for spread predictions.

4.5. Analysis of feature weights

As previously mentioned, we explored classification of the linear classifier with a set of basic features and a set of more complex partially derived from these basic features. The basic features included each team's

Elo score, average points scored, average points allowed, number of games played, number of games won, and win rate. The additional features we added included home court advantage (e.g. home court average points scored, home court average points allowed, home court win rate, etc.) as well as possessions, offensive rebounds, and defensive rebounds. In order to understand which features contributed most heavily to the win/loss and spreads predictions, we plotted the weights of each of the features. These relative weights are illustrated in Figure 3, in which the win rate features (away team win rate, home team win rate, away team home court win rate, and home team home court win rate) were in general the most important features. For three seasons of data however in win/loss predictions, the other features contributed much more relative to the classifiers for spreads predictions and one season of data.

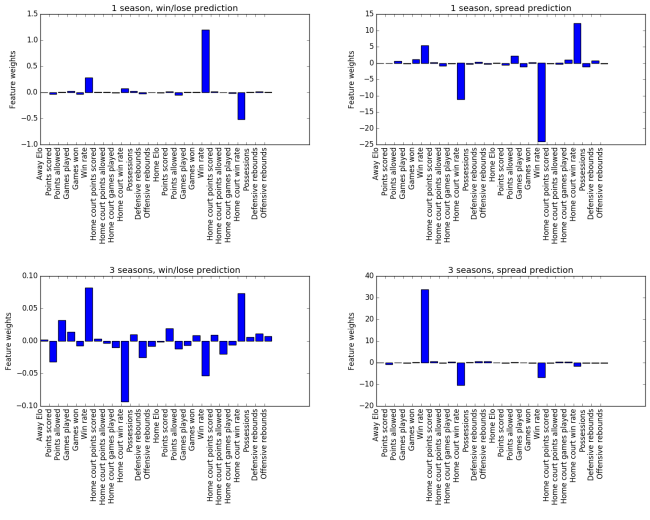

Figure 3. Linear classifier feature weights

Interestingly, when we train the linear classifier using only the win rate features, the classifier actually performs slightly better than with all the features. The numbers in Table 5 and 6 represent the accuracies and spread errors of the classifiers trained on all features and only win rate features when trained on one and three seasons of data, respectively. These findings seem to contradict the conclusions earlier in that adding more complex features such as home court advantage and rebounds improved the accuracy of the classifier. This could be due to overfitting to the number of features, especially in short periods of time in which averages might not necessarily be as representative of the overall skill level of a team.

	Win/Lose Accuracy	Spread Error
All features	66.9%	13.6
Only win rate	70.7%	8.95

Table 5. Classifier performance on 1 season

	Win/Lose Accuracy	Spread Error
All features	68.6%	9.29
Only win rate	68.7%	9.14

Table 6. Classifier performance on 3 seasons

4.6. Consideration of LASSO

When originally considering the types of models to implement and explore in trying to solve this problem, we had planned on using a LASSO model with feature weight sparsity in order to reduce overfitting. After visualizing the relative weights of the features in the linear classifier however, we decided that LASSO was not necessary in this case as the feature weights already exhibit significant sparsity for the large part. LASSO would likely therefore not have given significantly different feature weights and likely would not have significantly improved the accuracy or spread error of the classifier.

This could have been due to the skewed predictive powers of each of the different features. Intuitively, it is reasonable that certain features such as home and away team win rate would be much more important than other auxiliary features such as possessions or rebounds, which still provide some information, but to a much smaller degree. Thus, the natural structure of the problem and data lend the model towards sparse feature weights without the need for LASSO to force sparsity in the data and model.

5. Neural networks

After using a linear regression method to predict game outcomes as well as spreads, we next explored the use of neural networks for the very same problem. While we considered a multitude of neural networks at first, we ultimately decided to use a back-propagation multilayer perceptron network provided by `scikit-learn` for several reasons. First, the back-propagation aspect of the network allows us to perform supervised learning, which was our original goal. Secondly, the flexibility of this network made it very easy to change various hyperparameters such as the degree of regularization and the hidden layers size in order optimize for predictive capability. Finally, the network is able to perform both classification as well as regression, which enables

us to make predictions for the two forms of bets that we are interested in.

5.1. Design decisions

The ReLu function was the activation function of choice for the hidden layers because of sparsity as well as a reduced likelihood of vanishing gradients. For our weight optimization algorithm, we chose to use a batch gradient descent algorithm called the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm in stead of a stochastic gradient descent (SGD) algorithm. The rational behind this design decision was due to the relatively small size of our dataset. With only 300/ 900 samples for each phase of classification we didn't really need the speed offered by SGD and empirically, we found LBFGS to be much more accurate. Because the weights of the network are initialized randomly we decided to also utilize a consensus based approach. For classification, this means that each prediction we make is actually the mode of the outcome of 7 predictions by the same neural network. Thus, if the neural network predicts the home team wins in 5 instances, and that the away team wins in 2 instances, we would predict that the home team wins. For regression, this means that each spread prediction we make is actually the mean of the outcome of 7 spreads made by the same neural network.

5.2. Predicting win/loss

As mentioned previously, predicting the winner of a particular NBA game boils down to a classification problem.

5.2.1. Training

The input (X) to the multi-layer perceptron (MLP) classifier is a vector containing features for the two teams that are playing each other. At first, we utilized only the basic features but then we also incorporated the advanced features as well as just the win rate featurs to see if we could improve performance. The Y values given to the classifier were either 1 (if the home team wins) or 0. Additionally, we tested our classifier on both 1 season of data (2008-2009) as well as 3 seasons of data (2008-2011). We also tested both normalized and un-normalized data and found that normalization had no impact on the ultimate accuracy of the classifiers.

5.2.2. Validation

The purpose of validation is to determine the optimum hyperparameters for our neural network, which in our case is the regularization parameter as well as the size of the hidden layers. Through early testing/ experimentation it was discovered that hidden layers containing around 5-20 neurons performed the best so we incorporated that in creating our set of potential hidden layer sizes. Table 7 shows validation results for 3 seasons of data using both both basic and advanced features. The column headers are the hidden layer sizes (the i^{th} number represents the number of hidden neurons in the i^{th} hidden layer. The row headers are the various regularization coefficients we tried. We can see that using a regularization parameter of 0.0001 with one hidden layer consisting of 10 neurons yielded the best results Table 8 shows the aggregated validation results for classification. As a general trend, we see that the validation accuracy was almost always better with 3 seasons of data.

	(5,)	(10,)	(5,5)	(10,10)	(10,20,10)	(5,10,10,5)
0	0.684	0.671	0.665	0.683	0.654	0.671
0.0001	0.589	**0.687**	0.671	0.673	0.659	0.669
0.01	0.670	0.665	0.589	0.673	0.653	0.669
0.1	0.677	0.672	0.589	0.676	0.645	0.680
1	0.675	0.673	0.665	0.673	0.669	0.589

Table 7. Validation results for 2008-2011

	Basic features	Full features	Win rate features
2008	0.665	0.679	0.672
2008-2011	0.679	0.687	0.680

Table 8. Aggregate validation results

5.2.3. Testing

After finding the optimal architecture for our MLP classifier in each situation we proceeded to test the classifier on the testing data. The testing accuracies are depicted in Table 9.

We were able to get accuracies that were a bit higher than the previous ones found in literature (68 %). Even though using win rate features alone performed poorer than using the complete set of features during validation, it actually appears that using purely win rate features yielded a slightly higher accuracy during testing.

	Basic features	Full features	Win rate features
2008	0.660	0.672	0.687
2008-2011	0.671	0.693	0.695

Table 9. Aggregate testing results

5.2.4. Overfitting

When experimenting with various architectures for the neural network it was discovered that there was an interesting tradeoff between hidden layer complexity and testing accuracy, which highly resembled overfitting. We found that as we increased the number of hidden neurons in each hidden layer, we could increase training accuracy substantially but this ultimately resulted in extremely poor validation and testing accuracy.

5.3. Predicting spreads

As mentioned previously, predicting the spread of a particular NBA game boils down to a regression problem.

5.3.1. Training

The input (X) to the multi-layer perceptron (MLP) regressor is the exact same as the input to the classifier. The Y values, however, are now the actual realized spreads/ game outcomes (away team score - home team score). Just like for classification, we also tested our classifier on both 1 season of data as well as 3 seasons. Additionally, we also found that normalization had no impact on the performance of our predictor.

5.3.2. Validation

When finding the optimal architecture for our MLP regressor we used two different approaches for validation, which optimized for different things. The first approach involved optimizing the neural network to predict spreads that minimized the average prediction error for each game. For example, if our regressor predicts that the home team would win be 4 points, but in fact the away team wins by 2 points that results in an error of 6 points for that game. The second approach was a more direct approach towards our ultimate goal of beating the spread. It involves optimizing the neural network to specifically beat the spread. For example, if the spread favors the home team by 4 points, and our predictor predicts the home team wins by 6 points , and the actual home team wins by 8 points then this is considered "beating the spread". Ultimately, we wanted to choose the architecture that has the highest "beating the spread" rate. Unfortunately, after experimenting a bit, we saw that there was absolutely no difference between the two validation approaches so we just resorted to the first one. Table 10 shows validation results (with approach 1) for 3 seasons of data using both both basic and advanced features. Table 11 shows the aggregated validation results for validation.

	(5,)	**(10,)**	**(5,5)**
0	10.54	9.52	9.55
0.0001	9.56	9.49	9.56
0.01	9.55	9.50	9.45
0.1	9.47	9.55	9.50
1	9.52	09.51	10.21
	(10,10)	**(10,20,10)**	**(5,10,10,5)**
0	9.50	**9.44**	10.54
0.0001	9.51	9.51	9.46
0.01	9.51	10.54	9.46
0.1	9.53	9.50	9.45
1	9.49	9.58	9.48

Table 10. Validation results for 2008-2011

	Basic features	Full features	Win rate features
2008	9.84	9.55	9.48
2008-2011	9.60	9.44	9.21

Table 11. Aggregate validation results (approach 1)

We see here that using the win rate features with the 3 seasons of data yielded the best validation results. Interestingly, enough the architecture that performed the best in almost all cases consisted of a regularization value of 0.0001 and a hidden layer size of (10,20,10).

5.3.3. Testing

After finding the optimal architecture for our MLP regressor in each situation we proceeded to test the regressor on the testing data. The testing results are shown in Table 13. One of the interesting findings from these validation and testing results is that the predictor almost always does better with more data and that using either full features or win rate features consistently beats using basic features. With the additional data it is quite intuitive why this is the case. Regarding why full and win rate features outperform basic features our hypothesis is that the home court component plays a large part in the predictions because home court does not show up in the basic features at all.

	Basic features	Full features	Win rate features
2008	10.32	10.2	9.59
2008-2011	10.09	9.92	9.14

Table 12. Aggregate testing results (approach 1)

	Spread Error	LC Win rate	NN Win rate
A	8.70	48.2%	47.2%
B	8.65	46.6%	46.9%
C	8.67	48%	47.3%
D	8.67	47.8%	46.9%

Table 13. Comparison with other betting authorities

6. Evaluation with historical data

In addition to calculating and analyzing the average spread error, we also compared our spreads predictions against other betting authorities' spreads. After creating a scaper which scraped the spreads posted by other betting authorities, we were able to determine the rate at which our spread would have beaten the betting authority's spread. This would occur in two scenarios: (1) if our predicted spread indicates that the winning team would have won by more than the other betting authority's predicted spread (e.g. if our spread is -8 and the other betting authority's spread is -5 on a game in which the favored team won by more than five points), or (2) if the predicted spread indicates that the winning team would have won by less or lost compared with the other betting authority's predicted spread (e.g. if our spread is +2 and the other betting authority's spread is -5 on a game in which the favored team lost or won by less than five points).

6.1. Linear classifier

We compared our best linear classifier (only the four win rate features) against the spreads predicted by other betting authorities. We found that the spreads predicted by our linear classifier performed slightly worse than the spreads published by betting authorities. These results are summarized in Table 14, where win rate refers to the percentage of games our spread would have beaten the given betting authority's spread. Overall, the other betting authorities had an average spread error of approximately 8.5 points and would have beaten our spread a little more than 50% of the time. This is compared with our average spread error of 9.14 points for a linear classifier.

6.2. Neural networks

7. Evaluation with current data

In addition to evaluating our classifiers against historical data from previous seasons, we additionally explored evaluation against data from the current season. Practically, this would be the context in which we would use our classifier to predict the results and spreads of matchups which have not yet taken place. The input to the classifier is the running averages for the features so far in the season, and the output is either the predicted win/loss or the predicted spread on the matchup.

	Correct win/loss	Spread error
Charlotte @ Indiana	Correct	12.54
Milwaukee @ Toronto	Correct	14.64
Washington @ Miami	Wrong	11.30
Brooklyn @ Houston	Correct	7.82
Denver @ Dallas	Correct	18.40
Portland @ L.A. Clippers	Correct	6.39

Table 14. Performance on 12/12/2016 games

	Predicted	Other	Actual	Win?
Charlotte @ Indiana	-3.45	+1.5	-16	Yes
Milwaukee @ Toronto	-7.36	-8	-22	No
Washington @ Miami	+0.30	+1.5	-11	Yes
Brooklyn @ Houston	-11.8	-13.5	-4	Yes
Denver @ Dallas	-1.60	+2.5	-20	Yes
Portland @ L.A. Clippers	-7.39	-10	-1	Yes

Table 15. Spreads on 12/12/2016 games

7.1. Linear classifier

We trained our linear classifier on three seasons of past data and ran the classifier on some recent games this season. From this we were able to more realistically compare our win/loss predictions with the actual outcomes of current games and our spread predictions with other betting authorities' spreads and the actual spreads. Of the six games that took place on Monday 12/12/2016, the classifier was able to correctly predict the winner in five of the games. The average spread error on these games was 11.85 points with a standard deviation of 4.03 points. These results are given in Table 15, where the spread error represents the absolute value of the difference between the predicted spread and the actual spread. In comparison with our classifier and spreads, other betting authorities had an average spread error of 14.17 points with a standard deviation of 4.69 points.

When comparing our spreads with that of other betting authorities, we found that our classifier would have bet on the winning side of the spread in five of the six games. This is illustrated in Table 16 which compares the predicted spread, the other (betting authority's) spread, and the actual spread. These spreads are all given from the perspective of the home team,

	Regular	Conservative
Charlotte @ Indiana	Yes	Yes
Milwaukee @ Toronto	No	—
Washington @ Miami	Yes	Yes
Brooklyn @ Houston	Yes	Yes
Denver @ Dallas	Yes	Yes
Portland @ L.A. Clippers	Yes	Yes

Table 16. Comparison of different betting approaches

in which a negative spread indicates that the home team is favored to win by the given number of points. From this data, we can see that the spreads produced by the classifier would have led us to bet on the winning side of the betting authority's spreads in all of the games on Monday 12/12/2016 except for the matchup between Milwaukee Bucks and Toronto Raptors. This is indicated in the "Win?" column of the table.

Due to the arbitrage mentioned earlier in which you must bet more than $100 in order to win $100, we can also consider a slightly more conservative approach. More specifically, in the above model, we always bet on the spread regardless of the relationship between the predicted spread and the betting authority's spread. We could consider only betting if we are reasonably confident that our spread will beat the betting authority's spread. In other words, we could decide to bet only if our predicted spread is at least one point different from the betting authority's spread. If we were to adopt such an approach, then in the matchups for the games on Monday 12/12/2016, we would not have bet on one of the matchups and we would have won all of the spreads we bet on. These results are given in Table 17, in which the line for the Milwaukee and Toronto matchup in the conservative approach indicates that we would not have bet on that matchup.

Interestingly, we found that many of the other betting authorities had very similar spreads with only more noticeable deviations seen in the amount of money needed to bet in order to win $100. Because we are not considering how to place money when deciding on how to bet, our performance across the professional betting authorities was largely fairly consistent.

The fairly significantly different results between evaluating our classifier against other betting authorities on historical spreads and more current spreads are particularly interesting to us. This could be partially attributed to the fact that in the case of evaluating our classifier on current spreads, we train with all three seasons of data available to us, whereas when we test against historical data, we train on one season of data and set aside the other two seasons of data for validation and testing. Thus, this dramatic improvement in performance could be due to an increased number of training sample points. Either way however, such performance of our classifier makes us fairly confident in our ability to predict spreads when compared with professional betting authorities.

One matchup on Monday 12/12/2016 which is of particular interest here is the one between the Charlotte Hornets and Indiana Pacers. So far in the season, the Charlotte Hornets had 14-10 record whereas the Indiana Pacers had 12-12 record. Based on win rate alone, one might think that the Hornets would have had the edge in this game. If we look at the home court records however, we can see that the Pacers, who were playing at home, had a home court record of 9-4 while the Hornets had a road record of 6-4. Thus, by considering the home court advantage, we were able to correctly predict that Indiana Pacers would win and therefore hypothetically bet on the winning side of professional betting authority spreads.

7.2. Neural networks

8. Potential sources of failure

Some drawbacks to the algorithm presented here include the fact that we do not examine individual player data. This was a choice made largely due to the fact that individualized player data is quite difficult to obtain, especially in quantities large enough for accurate training, validation, and testing. Because we look at a team's performance overall so far in the season however, this means that our algorithm is susceptible to changes in starting player lineups, player trades made during the season, and player injuries. Thus, despite the acceptable performance of our algorithm, we do identify this source of weakness in the ability to predict spreads and classify win/loss in matchups. Future iterations of the algorithm would therefore primarily seek to address these potential sources of failure in order to further improve and refine the overall predictive power of the algorithm.

Future iterations of this algorithm would explore the addition of features such as the number of assists, steals, blocks, turnovers, and fouls per game, as well as assist to turnover ratio and average player efficiency ratings.

9. Conclusion

In this report, we have explored the use of supervised machine learning techniques to predict win/loss and

spreads on basketball matchups in the NBA. We primarily explored the use of linear classifiers and neural networks in our approach, where we explored various levels of regularization, architectures, features, and seasons of data. We found that overall the best polynomial basis classifier was a linear classifier which only took win rate features (home/away team win rate and home/away team home court win rate) into account. This classifier was able to achieve about 70% accuracy in predicting win/loss and had an average spread error of 8.95 points when compared with the actual spread. We found that for the neural networks, classification of win/loss was best with greater regularization and fewer hidden layers (an optimal architecture of (10) and $\alpha = 0.1$), whereas regression was optimal with a smaller regularization term and more hidden layers (an optimal architecture of (10,20,10) and $\alpha = 0.0001$). Overall, we were unable to reliably beat the spreads published professional betting organizations when analyzing historical data. When examining game matchups from the current season however, based on our limited access to data, we found that our classifiers did indeed do a remarkable job of outperforming and winning against the professional spreads, especially when we adopted a more conservative approach.

10. Breakdown of individual work

The breakdown of work between the two members of the group largely followed that given in the original project proposal. Andrea and Tyson jointly worked on preliminary review of the current techniques and the state of the art. Following this, Tyson was primarily involved in the collection of game data from past seasons, and Andrea created the scraper to collect data on historical spreads put out by various betting organizations on previous games. Andrea explored preliminary models such as linear regression and regression with various basis functions, and Tyson chiefly explored a neural network approach. As discussed during the project proposal meeting, we decided not to implement a convolutional neural network for this problem, as the architecture did not fit the structure of the problem as closely. We each tested the models we worked on against data test sets, historical spreads, and games in the current season. We split the writing up of the report equally, where Andrea wrote part of the data collection, linear classifier, linear classifier evaluation, and potential sources of failure sections, and Tyson wrote up the introduction, part of the data collection, neural network, and neural network evaluation sections.