Predicting NBA Game Outcomes using Machine Learning

Fall 2021, COM S 574 Final Project

Team Members

Kaden Kilburg (S E/Senior): kadenk@iastate.edu
Murtaza Zohair (M E/Ph.D.): mzohair@iastate.edu
Seongyoung Kong (CHEM/Ph.D.): sykong@iastate.edu
Yongsuk Cho (M E/Ph.D.): yscho@iastate.edu
Youngro Lee (AER E/Ph.D.): youngro@iastate.edu

December 14, 2021

ABSTRACT

This report considered the problem of predicting NBA game outcomes. It considers both linear model approaches as well as those involved in deep learning. Game predictions made in this report cover both point spread between team scores and game outcome, which correspond to which team would win given their player statistics. We find that our linear regression models performed at a 65% accuracy with a 0.63 F1 score for classification problems, while the deep learning models approached 67% accuracy with a similar 0.67 F1 score.

INTRODUCTION

The National Basketball Association (NBA) is the top basketball league in the world, and people love to watch these games. According to sports popularity research done by Miller et al., The NBA was ranked in no. 1 sport in the world in 2017. Due to its popularity, the professional basketball market has become a billion-dollar business. Tons of new talented players have been joining the league every new season, and player trades have been actively happening between teams. To survive in such a big market, every team struggles to bring up the best offense/defense strategies and have the best roster with a limited budget every year.

In order to fully utilize each player's potential, the coaching staff is always playing with numbers. They review the game, so-called a video review, before/after each game. Nowadays, there are more advanced technologies available to get more data, such as individual players' moving paths, etc. Most NBA teams are now investing tons of money in the data science area to analyze this vast in-game data and improve the chance of winning. Recently, there have been various types of research done with the aid of machine learning. Thabtah et al. worked on the feature analysis using both linear/nonlinear supervised machine learning techniques to see which game stats contribute to predicting the game result. They reported that the defensive rebound feature was the most important parameter influencing the game result. Three-point percentage, free throws made, and total rebounds followed the next crucial factors accordingly. Beckler et al. further analyzed the data to bring up better performance indicators with having a linear combination of features.

As the result of various statistical analyses, the NBA's play style has been evolving dramatically in the last decades. The basketball game-winning strategies have been changed so much. For example, the offensive rating and defensive rating are decent indicators to track a player's performance. These are the combination of multiple features. A higher total rebound and assist/turnover ratio lead to good team performance. Above mentioned parameters are time-invariant features, meaning the contribution ratio to winning has been barely changed historically. For example, a higher assist/turnover ratio has almost always led to predicting winning the game. On the other hand, there are time-variant (or season variant) features as well. With the aid of data science, several NBA teams, including Houston Rockets and Golden State Warriors, realized shooting more threes contributes to a higher chance of winning. They analyzed that as a result of shooting more 3pt shots, players get more space, which leads to offense easier and makes defense harder. Another trend is Pace. According to Curcic, the average weight and height of NBA players have decreased in the last decade. This result reflects today's NBA play strategy. Each team tries to have more transition offense at a fast pace.

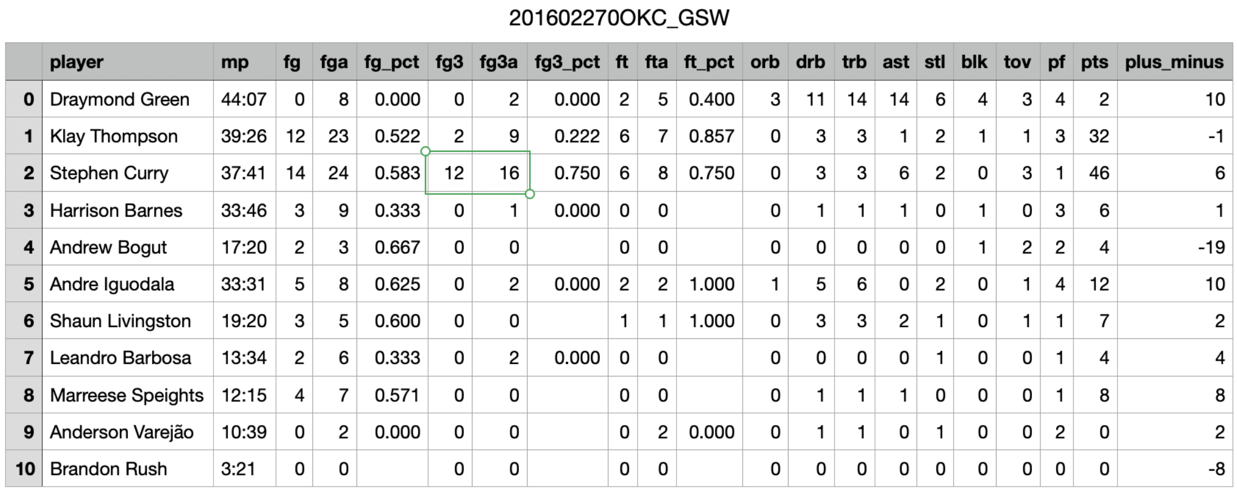
The NBA season is split into two periods: the regular season and the playoffs. The regular season consists of 82 games (the 2020-21 season was shortened to 72 games). The win totals within a team's conference are utilized for the selection and seeding of the playoffs. Only teams in the playoffs qualify for a chance at the championship game, and higher seeding has advantages for home games and the strength of their schedule in the playoffs. Typical analysis of NBA player and team performance relies on box score metrics such as points, rebounds, field goal percentage (FG%), and etc. The raw values can be misleading, but they clearly have a relationship with the result of NBA games as the team with the most points wins a game. This project aims to use machine learning to evaluate the relationship between these various metrics and the overall game result to predict the outcome of NBA games. In that context, we would like to predict each team's performance, such as the overall winning rate of a team, specific winning chance against a specific team, and final rank. We will use the following data set: players' stats, team stats, playing at home or not, weather, etc. using several statistical machine learning methods.

The rest of this report is organized as follows: What kind of data of NBA players will be used is addressed in section 2. Section 3 explains statistical machine learning methods that will be used to predict the winning rate of each player. Section 4 presents machine learning results depending on an adopted method. Lastly, Section 5 concludes this report.

DATA PREPARATION

For the data acquisition, a python web-scraper was built to collect all previous NBA games beginning from the 1982-1983 season to the 2020-2021 season. All the data was acquired from *basketball-reference.com*, where basketball data is freely distributed for various research purposes. The data collected consists of various statistics divided between seasons. For each season, a dataset of all games played in that season between teams including the points scored by each team, several datasets consisting of all game-specific player stats (one dataset per team per game), another dataset consisting of "Per 36 Minutes" seasonal statistics for each rostered player within the NBA, and several more datasets for each team which consist of the roster information.

As an example, Table 1 shows a game-specific player stats dataset corresponding to one of the participating teams. This specific game was the Oklahoma City Thunders' home game against the Golden State Warriors held on February 27th, 2016. The table headers shown are variable names that can be mapped to their true meaning in Table 2. Another dataset is normalized to 36 mins stat per game. We appended "_per_g" to differentiate between datasets.


201602270OKC_GSW

	player	mp	fg	fga	fg_pct	fg3	fg3a	fg3_pct	ft	fta	ft_pct	orb	drb	trb	ast	stl	blk	tov	pf	pts	plus_minus
0	Draymond Green	44:07	0	8	0.000	0	2	0.000	2	5	0.400	3	11	14	14	6	4	3	4	2	10
1	Klay Thompson	39:26	12	23	0.522	2	9	0.222	6	7	0.857	0	3	3	1	2	1	1	3	32	-1
2	Stephen Curry	37:41	14	24	0.583	12	16	0.750	6	8	0.750	0	3	3	6	2	0	3	1	46	6
3	Harrison Barnes	33:46	3	9	0.333	0	1	0.000	0	0		0	1	1	1	0	1	0	3	6	1
4	Andrew Bogut	17:20	2	3	0.667	0	0		0	0		0	0	0	0	0	1	2	2	4	-19
5	Andre Iguodala	33:31	5	8	0.625	0	2	0.000	2	2	1.000	1	5	6	0	2	0	1	4	12	10
6	Shaun Livingston	19:20	3	5	0.600	0	0		1	1	1.000	0	3	3	2	1	0	1	1	7	2
7	Leandro Barbosa	13:34	2	6	0.333	0	2	0.000	0	0		0	0	0	0	1	0	0	1	4	4
8	Marreese Speights	12:15	4	7	0.571	0	0		0	0		0	1	1	1	0	0	0	1	8	8
9	Anderson Varejão	10:39	0	2	0.000	0	0		0	2	0.000	0	1	1	0	1	0	0	2	0	2
10	Brandon Rush	3:21	0	0		0	0		0	0		0	0	0	0	0	0	0	0	0	-8

Table 1. Single team game-specific player dataset example: Golden State Warriors' player stats at Oklahoma City held on February 27th, 2016.

Table Header	Meaning
player	Player Name
mp	Minutes Played
fg	Field Goals
fga	Field Goal Attempts
fg_pct	Field Goal Percentage

fg3	3-Point Field Goals
fg3a	3-Point Field Goal Attempts
fg3_pct	3-Point Field Goal Percentage
fg2	2-Point Field Goals
fg2a	2-Point Field Goal Attempts
fg2_pct	2-Point Field Goal Percentage
efg_pct	Effective Field Goal Percentage
ft	Free Throws
fta	Free Throw Attempts
ft_pct	Free Throw Percentage
orb	Offensive Rebounds
drb	Defensive Rebounds
trb	Total Rebounds
ast	Assists
stl	Steals
blk	Blocks
tov	Turnovers
pf	Personal Fouls
pts	Points

Table 2. Table header meanings for a game-specific player dataset.

MODEL SELECTION

Within this project, there are two primary categories of predictions we performed for our NBA datasets. The categories are those predictions that involve linear models, as taught in class, and deep neural network models. The linear model section is fairly straightforward and does not require much background knowledge to understand. The neural network section is slightly more complicated, and it may help to have experience with similar neural network tools such as Tensorflow's Keras API as used in this project.

Linear Models (OLS, Ridge, Lasso, and Elastic Net)

Large dataset with many features that have an unknown relationship to one another. Linear regression models allow us to predict the game outcome through point differential but also give insight into how each feature impacts the game outcome in an easily interpretable way. The following linear models are a few of which our team tested for this report.

The first method considered was Ordinary Least Squares (OLS). OLS is the most commonly used method to fit a linear model. We want to find coefficients such that

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

best estimates the true *Y* where *Y*-hat is a response, X_i is a feature, and β_i is a coefficient corresponding to feature X_i. OLS finds the coefficients that minimize its loss function, the residual sum of squares (RSS), which is defined as

$$RSS = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Usually, OLS tends to find the coefficients optimistically such that a training error is much lower than testing error.

The second linear model considered is ridge regression. Ridge is very similar to OLS in that its loss function is a natural extension of RSS:

$$RSS + \lambda \sum_{i=1}^{n} \beta_i^2$$

where $\lambda \geq 0$ is a tuning parameter that is determined separately. The term following the RSS in the above equation is called a shrinkage penalty. As λ becomes larger, the shrinkage penalty has a more significant impact on the equation causing coefficients to approach zero. An optimal λ value is one that prevents overfitting while minimizing test error. Cross-validation is useful for finding such a λ.

Lasso regression is the third linear model tested with our datasets. Lasso is similar to Ridge in how it extends OLS; however, their shrinkage penalties differ. Lasso's shrinkage penalty changes from Ridge by removing the squared power from the coefficients and, instead, taking their absolute values.

$$RSS + \lambda \sum_{i=1}^{n} |\beta_i|$$

One impact aspect of Lasso is that it can force some of the coefficients to be exactly zero when λ is large enough. This allows the elimination of less useful features in a fitted model by dropping their coefficients to zero. This differs from Ridge, which can only force its coefficients to approach, but not equal, zero.

Finally, the last linear model considered is Elastic Net which combines aspects of both Lasso and Ridge. That is, Elastic Net adds both of their shrinkage penalties to the RSS:

$$RSS + \lambda_1 \sum_{i=1}^{n} |\beta_i| + \lambda_2 \sum_{i=1}^{n} \beta_i^2$$

This lets the loss function become convex by the existence of the quadratic penalty term of Ridge while still retaining Lasso's feature selection ability.

With the chosen linear models explained, a couple of tested terms will now be explained. These are Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). RMSE is a frequently used measure of the differences between the values observed and the value predicted by a model fitting. OLS finds the coefficients of the features minimizing RMSE; this is the same as minimizing the RSS.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

MAPE is another measure that is frequently used in statistical analysis. Its formula is given by

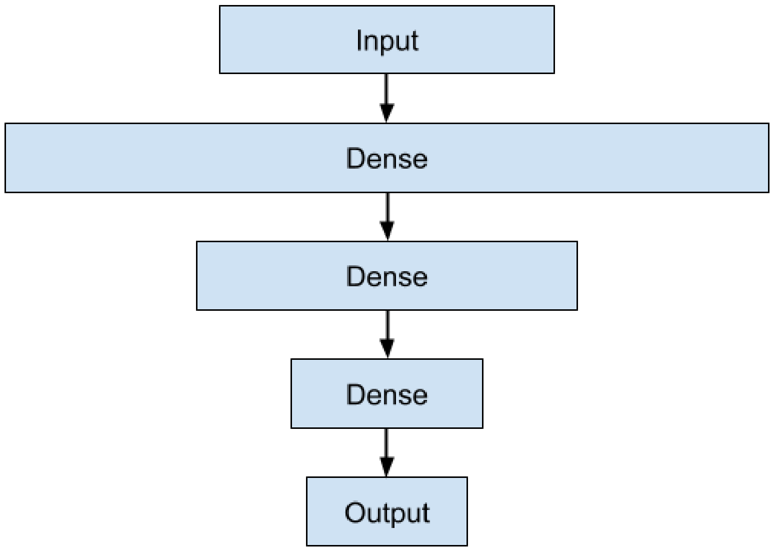
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|$$

where Y_i is the true value for the *i*th sample, and *Y*-hat$_i$ is its predicted value.

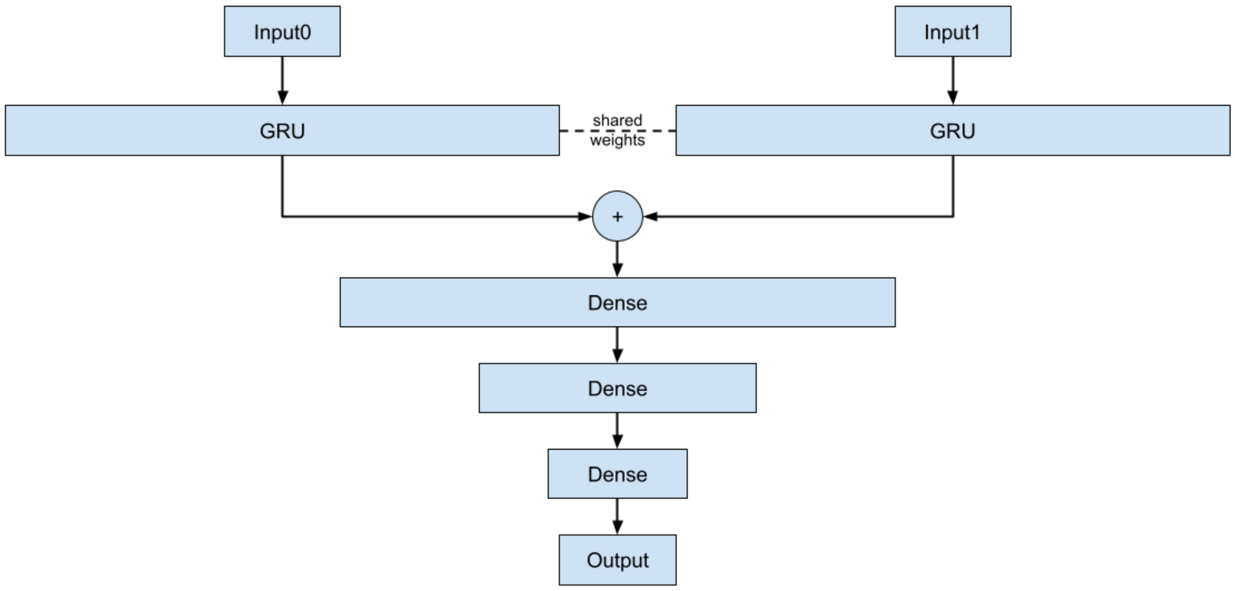
For all models, the data was split 80/20 into training and validation sets. 4-fold cross-validation is performed on the training data for Ridge, Lasso, and Elastic Net to find the optimal parameters. The test RMSE and MAPE are recorded on the validation set. This process is repeated for 5 runs, and the average error values over 5 runs are reported. 15 values of alpha between 1E-7 and 1E3 are tested. 5 values of the l_1 ratio between 1E-4 and 0.9999 are tested for Elastic Net. The model coefficients using the optimal parameters are recorded below. The predicted and true values of point differentials for all games are plotted below in the "Results" section.

Deep Neural Networks

Deep neural networks are a common machine learning approach in today's world. Their ability to extract secondary features from a provided dataset and model highly complex behaviors and relationships make them a preferred method by many researchers and programmers. This powerful predictive ability comes with notable flaws such as requiring high computational power, requiring large data sets, hiding any true relationships within the many weights of the neural network, and lacking a clear application procedure (RF Wireless World, n.d.). However, these first two points mentioned are not a concern for our project as modern computers possess sufficient computational power, and our dataset is sufficiently large for our chosen neural network structures.



(a) A sequential model with hidden dense layers of size 128, 64, and 32 neurons, respectively. It accepts a 1D input vector and outputs a single value.



(b) A functional model making use of a shared GRU layer. This results in mapping the 2D input to a 1D vector to feed into layers identical to those in (a).

Figure 1. (a) and (b) show the two neural network models used for predictions in this report.

We have chosen to test two distinct network structures for this project, which can be seen in Figure 1, where both were constructed using the Tensorflow Keras API. These networks are both used to predict a winning team of a game, given all the players for each team.

The first network in Figure 1a begins by taking the average seasonal player statistics within a team and concatenating these averaged features with the other team's averaged features. Therefore, the first half of features correspond to the visitor team, while the second half of features correspond to the home team producing a feature set of size $p = 44$ (22 for each team). This network is a sequential model consisting of five layers: the input layer, three dense layers using 128, 64, and 32 neurons, respectively, and where each layer is activated by a RELU function, and finally, it concludes with an output layer of size one.

The second network shown in Figure 1b begins by constructing a separate two-dimensional matrix of the seasonal player statistics within a team for both teams; note that this differs from Figure 1a, which finds the average team statistics to fit within a one-dimensional vector. Then for Figure 1b, each team matrix is fed into a separate input of a functional neural network. These inputs are then treated as time series data and fed through a shared GRU layer with 128 neurons. This step's purpose is similar to the team averaging step taken in the previous figure, which flattens the team statistics into a one-dimensional vector. However, this GRU layer allows the neural network to discover its own flattening function, which may prove better than taking the average. Note that the team matrix could not be flattened directly by reshaping the matrix into a vector for two reasons: one, this would result in far too many features, and two, this doesn't allow for teams of different sizes. Once both teams are flattened by the recursive layer, these outputs are concatenated and sequentially fed through a model identical to that of Figure 1a.

RESULTS

Using a series of statistical operations and joins between the several datasets collected, we constructed two new datasets to train with. The first dataset was used for training both the linear models and one of the deep neural network models. This dataset defines each of its samples to represent a game. Each sample only contains the number of points scored by both the visitor team and the home team, along with the concatenated sets of each team's averaged seasonal player statistics for those who had game-time within the sample game. Therefore, because each player has 22 features to use, this resulting dataset consists of 44. The second dataset is used solely by the 2D input neural network model and holds a similar shape to the first dataset; however, the concatenated team feature sets are unaveraged and consists of 2D matrices each of size $P \; x \; M_i$ where $P = 22$ and is the number of seasonal player statistic features used while M_i varies with how many players from team i had game-time.

Following the construction of each dataset, the was a small amount of preprocessing performed on the data to produce better results. The step was to balance the dataset by stripping samples to leave an equal number of home and visitor wins. The next step was to normalize the dataset by interpolating all features to a scale of 0 to 1 where 0 represents the smallest value found for that feature in the dataset and

1 represents the largest value.

Linear Models:

The results from various linear regression models are presented below. All of the models have similar performance, with an average RMSE of about 12, which corresponds to a prediction error on point differentials of around 12 points. Lasso and Elastic Net are both able to perform feature selection, but both models choose the lowest value of alpha so that the solution is close to the OLS solution and all features are used. This indicates that all the features have large independent contributions to the point differential prediction, but this is counterintuitive considering features like field goal percentage, which is the ratio of the first two features field goal made and field goal attempts. The models all underpredict game outcomes for the outlier games with high point differentials.

Model	MAPE	Run RMSE	Optimal Parameters
OLS	1.11	12.1	N/A
Ridge	1.13	12.1	$\lambda = 0.0450$ All features used
Lasso	1.304	12.17	λ =1E-7, All features used
Elastic Net	1.122	12.12	λ_1 =1E-7, λ_2 = 1E-4, All features used

Table 3. Results from various linear regression models

Figures 2-5 show the predicted versus the actual values depend on the adopted linear regression method. If the points are located similarly on both axes and distributed in a linear form, it means that the error is small since the predicted and actual values would be very similar. In our case, the errors are around 12 RMSE, so the prediction is off on average by 12 points of the actual observation. Additionally, Tables 4-7 show the confusion matrices for those same figures by observing the sign of the linear regression predictions to classify whether or not the home team won.

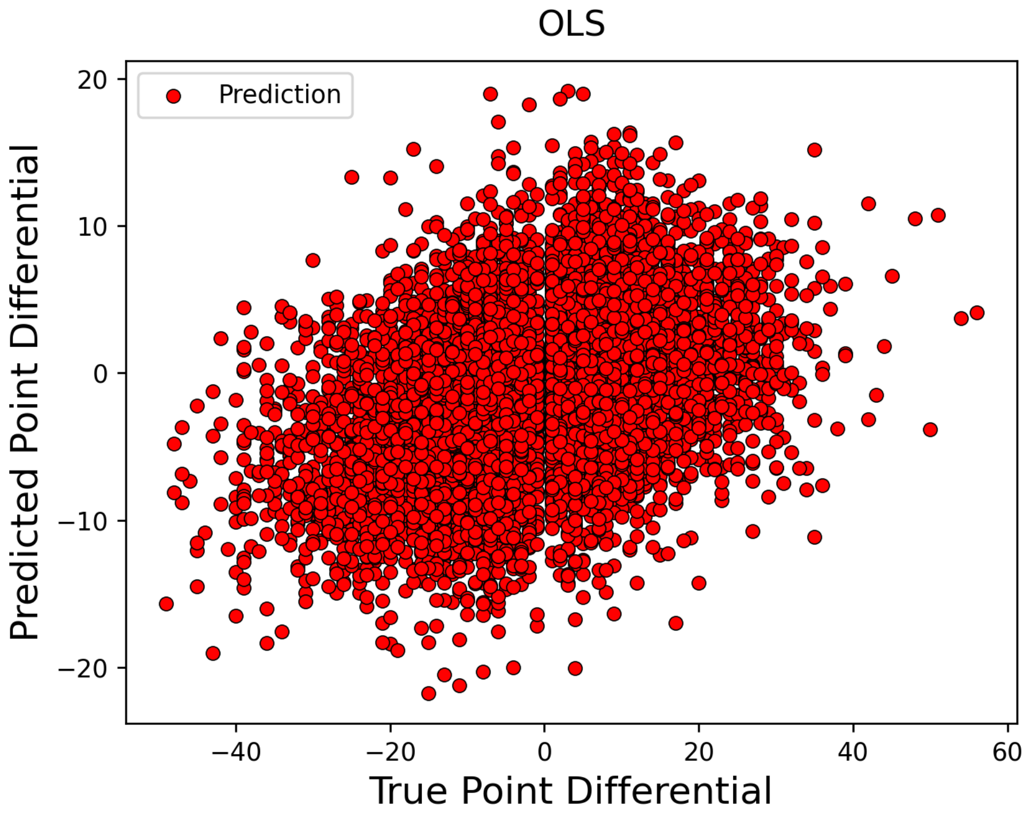


Figure 2. Prediction results of OLS

		True	
		Positive	Negative
Predicted	Positive	2534	1770
	Negative	1251	3110

Table 4. Confusion matrix using OLS predictions. Accuracy ≈ 65.1% with precision ≈ 0.59, recall ≈ 0.67, and F1 score ≈ 0.63.

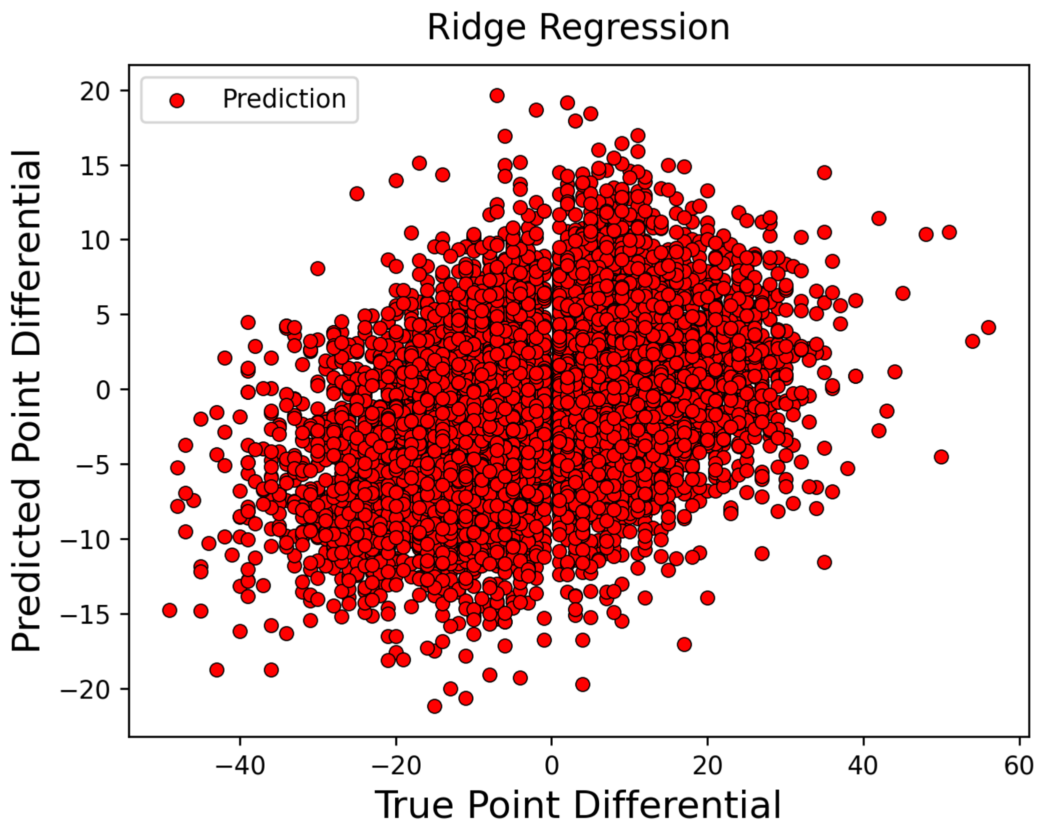


Figure 3. Prediction results of Ridge

		True	
		Positive	Negative
Predicted	Positive	2560	1744
	Negative	1259	3102

Table 5. Confusion matrix using Ridge predictions. Accuracy ≈ 65.3% with precision ≈ 0.59, recall ≈ 0.67, and F1 score ≈ 0.63.

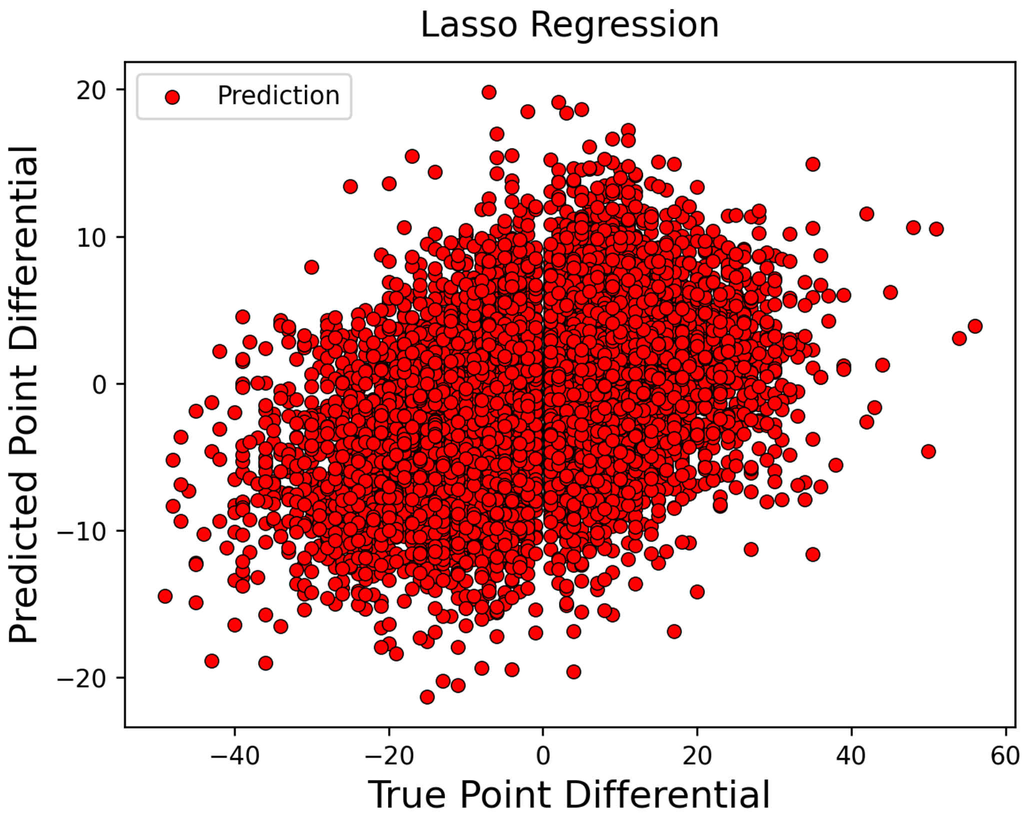


Figure 4. Prediction results of Lasso

		True	
		Positive	Negative
Predicted	Positive	2557	1747
	Negative	1258	3103

Table 6. Confusion matrix using Lasso predictions. Accuracy ≈ 65.3% with precision ≈ 0.59, recall ≈ 0.67, and F1 score ≈ 0.63.

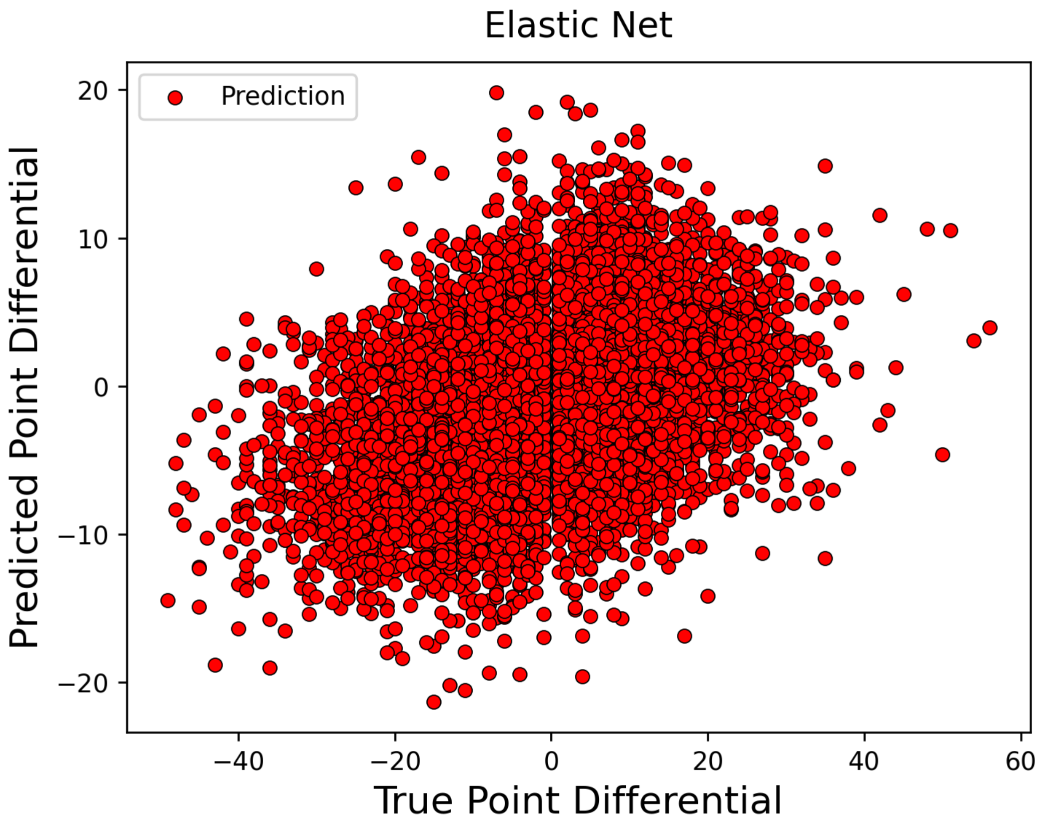


Figure 5. Prediction results of Elastic Net

		True	
		Positive	Negative
Predicted	Positive	2557	1747
	Negative	1257	3104

Table 7. Confusion matrix using Elastic Net predictions. Accuracy ≈ 65.3% with precision ≈ 0.59, recall ≈ 0.67, and F1 score ≈ 0.63.

The determined coefficients of each linear model are listed in Table 8. The base meaning of each coefficient's feature is given in Table 2; however, the features used here are the average seasonal statistics for the players who participated in a given game as opposed to game-specific player stats. Note that 1 and 0 following after each feature mean home and away, respectively. One can notice that all four methods have different coefficients on each feature, but their overall tendencies are quite similar. For example, in the case of field goals per 36 minutes (fg_per_g) in the visitor match, OLS and Ridge have positive coefficients, whereas Lasso and Elastic net have negative coefficients. Although one can intuitively think that a high field goal score would lead to a high winning rate, the Lasso and Elastic net results are counterintuitive. The most affecting feature on the winning rate is 2-point field goals (fg2_per_g0), and the least affecting feature on the winning rate is turnovers (tov_per_g0) for the visitor match.

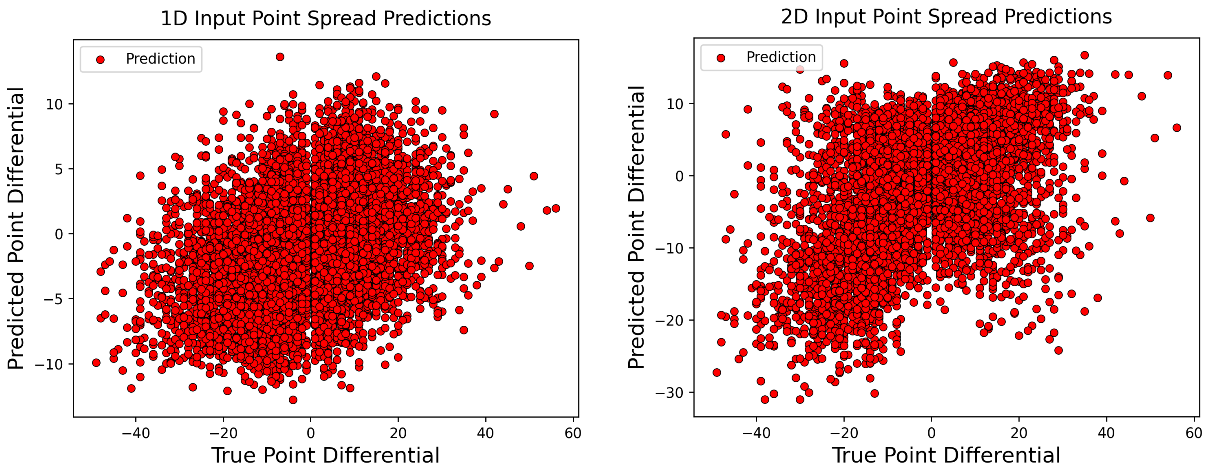
Features	OLS	Lasso	Ridge	Elastic net	Features	OLS	Lasso	Ridge	Elastic net
fg_per_g0	-4.59	-24.53	4.49	-23.63	fg_per_g1	-15.35	25.63	-13.84	24.1
fga_per_g0	1.45	-47.93	-33.83	-46.66	fga_per_g1	-58.82	27.42	47.90	29.31
fg_pct0	-17.73	-15.02	-13.56	-15.02	fg_pct1	12.47	18.13	16.30	18.11
fg3_per_g0	48.93	52.60	28.38	51.99	fg3_per_g1	-53.35	-56.46	-43.82	-56.39
fg3a_per_g0	-54.10	-21.57	-26.94	-22.36	fg3a_per_g1	120.32	70.31	48.76	68.97
fg3_pct0	1.46	1.03	1.08	1.03	fg3_pct1	3.25	2.62	2.62	2.62
fg2_per_g0	41.16	55.76	22.43	54.84	fg2_per_g1	-27.89	-37.08	-18.21	-36.6
fg2a_per_g0	-44.18	-7.46	-18.14	-8.5	fg2a_per_g1	113.73	42.68	24.27	41.07
fg2_pct0	3.26	0.34	0.00	0.34	fg2_pct1	-6.68	-9.7	-8.35	-9.68
efg_pct0	33.55	30.67	29.49	30.66	efg_pct1	-7.86	-8.65	-8.69	-8.65
ft_per_g0	34.3	27.70	23.77	27.64	ft_per_g1	-25.66	-17.90	-22.89	-18.21
fta_per_g0	-17.38	-14.56	-13.86	-14.57	fta_per_g1	13.82	16.02	15.20	16.01
ft_pct0	-1.72	-1.41	-1.23	-1.41	ft_pct1	2.34	2.82	2.61	2.81
orb_per_g0	-0.61	-3.69	1.91	-3.53	orb_per_g1	-3.86	11.21	0.70	10.94
drb_per_g0	-4.47	-7.01	5.98	-6.66	drb_per_g1	-5.24	23.64	2.69	23.11
trb_per_g0	27.98	33.2	16.53	32.75	trb_per_g1	-6.17	-47.2	-18.10	-46.46
ast_per_g0	10.6	10.90	10.95	10.91	ast_per_g1	-12.85	-11.72	-11.92	-11.72
stl_per_g0	6.60	7.82	7.68	7.82	stl_per_g1	-8.42	-9.76	-9.46	-9.75
blk_per_g0	2.63	2.97	2.96	2.97	blk_per_g1	-3.93	-4.43	-4.48	-4.43
tov_per_g0	-13.19	-13.93	-14.09	-13.93	tov_per_g1	19.15	18.68	18.86	18.68
pf_per_g0	-10.75	-10.46	-10.55	-10.46	pf_per_g1	11.24	11.03	11.01	11.03
pts_per_g0	-11.71	-2.35	12.52	-2.03	pts_per_g1	-14.62	-56.56	-32.13	-55.24

Table 8. Determined Coefficients of the Linear Model Results

Deep Neural Networks:

This section lists the results achieved after training the two neural networks specified in Figure 1. Three variations of each neural network were trained. For each variation, their prediction results are highlighted in Table 5, which displays the confusion matrices for each model's variations. Note that the model displayed in Figure 1a is referred to as the 1D input model, while the model displayed in Figure 1b is referred to as the 2D input model. The datasets used here were split 60/40 for training and testing data. Furthermore, 10% of the training data was split for use as a validation set during training.

The first variation trained both neural networks using the Mean Squared Error (MSE) loss function. Additionally, the model was trained using each game's point spread, or difference in team's points. The test set predictions can be seen in Figure 6, which plots each sample's true point spread on the horizontal axis with that sample's predicted point spread on the vertical axis. Note that negative values correspond to the home team winning that game, while positive values correspond to the visitor team winning. Therefore, using the model's regressive output, we can convert the model to a classification model by analyzing the sign of the predictive point spread. These classification results are displayed in the confusion matrix of Table 5a.



(a) 1D Input Results **(b)** 2D Input Results

Figure 6. The plots shown above give the spread predictions produced by the two proposed models trained using MSE.

		True	
		Positive	Negative
Predicted w/ 1D Input	Positive	2378	1215
	Negative	1926	3146
Predicted w/ 2D Input	Positive	2890	1432
	Negative	1414	2929

(a) Regression neural network using MSE loss function: For the 1D model, it has an accuracy ≈ 63.8% with precision ≈ 0.66, recall ≈ 0.55, and F1 score ≈ 0.60. For the 2D model, it has an accuracy ≈ 67.1% with precision ≈ 0.67, recall ≈ 0.67, and F1 score ≈ 0.67.

		True	
		Positive	Negative
Predicted w/ 1D Input	Positive	2531	1396
	Negative	1773	2965
Predicted w/ 2D Input	Positive	2996	1550
	Negative	1308	2811

(b) Regression neural network using MAE loss function: For the 1D model, it has an accuracy ≈ 63.4% with precision ≈ 0.64, recall ≈ 0.59, and F1 score ≈ 0.61. For the 2D model, it has an accuracy ≈ 67.0% with precision ≈ 0.66, recall ≈ 0.70, and F1 score ≈ 0.68.

		True	
		Positive	Negative
Predicted w/ 1D Input	Positive	2784	1480
	Negative	1577	2824
Predicted w/ 2D Input	Positive	2660	1174
	Negative	1701	3130

(c) Classification neural net using binary cross-entropy loss function: For the 1D model, it has an accuracy ≈ 64.7% with precision ≈ 0.65, recall ≈ 0.64, and F1 score ≈ 0.65. For the 2D model, it has an accuracy ≈ 66.8% with precision ≈ 0.69, recall ≈ 0.61, and F1 score ≈ 0.65.

Table 5. The tables shown above are each the joined confusion matrices for the two models displayed in Figure 1. (a) and (b) use a neural network which first estimates the spread of a game as shown in Figure 6. By checking the sign of the predicted point difference, this is then used for classifying the game's winner. (c) show the results of modifying the output layer to use a sigmoid function and thresholding at 0.5 to classify a winner.

The second variation of the trained models used Mean Absolute Error (MAE) as the loss function. This is in contrast with the first variation, which used MSE. The regression plots are not displayed in this report as they appear similar to the MSE graphs. However, this variation was also used to indirectly classify the winner of each test game. The confusion matrix results are shown in Table 5b. Compared with the MSE loss function results, this MAE variation performs similarly to MSE.

The third variation sets itself apart from the first two. Due to the first two variations being trained using and to predict point spread directly, they could only indirectly predict which team won a game.

However, this third variation was directly trained to predict the winning team. That is, the training dataset was modified beforehand such that the only labels were zero or one. Zero corresponds to the visitor team winning, while one corresponds to the home team winning. Furthermore, the output layer of each neural network was given a sigmoid activation function such that all predictions would occur within the range of zero to one. Then, a threshold set at 0.5 was used to determine predicted winners: predictions less than or equal to 0.5 would assign the winning team to the visitors, while predictions greater than 0.5 would result in the opposite. The models used were trained using a binary cross-entropy loss function. This loss function works similarly to MSE in the sense that it penalizes error more the greater the error is; however, it works specifically within the range of zero to one and with only two labels. The test set results are shown in the confusion matrix of Table 5c.

As visible from Table 5, the third variation using direct classification performs better than its indirect counterparts while using the 1D input model; however, the indirect classifications perform better for the tested 2D input model. These performance comparisons were based on their F1 scores.

DISCUSSION

Our work in this project has produced valuable results which should be taken into account when performing NBA predictions. One of the main points is that the chosen linear models perform poorly at classifying whether or not a home team will win given the tested feature set. However, the deep learning approach shows significant promise with 67% accuracy when used to classify samples directly. Furthermore, the accuracy is reinforced by the even distribution between precision and recall scores, as seen by the F1 score also reaching 0.67 in the best model.

On the other hand, the linear regression models performed fairly well with a best accuracy and F1 score of 65.3% and 0.63, respectively. Similar models may perform better if used to directly classify winners versus the indirect point spread approach taken in the project; however, this may prove false, as seen from the indirect approach outperforming the direct approach while using deep learning.

Another aspect researched in this project was the performance difference between the 1D and 2D input neural networks. It is clear from each variation's larger F1 score that the 2D input model performs better than that of the 1D input. However, although it was not reported, it should be noted that, due to its recurrent GRU layer, the 2D input model trains at a significantly slower pace than the 1D model. Furthermore, the preprocessing step required for data into the 2D input model takes a fairly large amount of time as well in order to ensure the separate team matrices are properly fed into the network. Due to this, it may be reasonable to use the less-predictive 1D model in favor of execution time.

Before leaving the subject of the GRU layer, a comparison should be made between the plots displayed in Figure 6. The 1D, as it was trained using the same input as the linear models, shows an expected radial structure. However, the 2D input model shows an interesting behavior that appears to avoid predicting visitor wins when, in reality, the home team won. This proves there is some underlying relationship within our dataset that can improve the precision or recall, likely both, of our predictions. Evidence of this relationship was only seen in the 2D input model, which fits with the assumption that an overall team composition has more predictive power than average team statistics. While the GRU layer is unlikely to be the best choice for converting the 2D input into a 1D input for use in the following layers, it has shown promise, as stated above.

One of the most important factors which contributed to the success of the deep learning models was the balancing and normalization of the training and testing set. Before these steps were performed, the models would perform poorly by always predicting the same winner classification in the direct approach.

Overall, our project has shown substantial results; however, there still exist many improvements and areas of research for further continuation of our work. One path to consider is better feature engineering to possibly improve the linear models as well as those used for deep learning. Another path is to experiment with similar 2D input models to try and produce more accurate predictions. For a final path, we suggest testing with the last season's seasonal data and opposed to the current season's post seasonal data. This is because the work shown in this report is not generalizable to future predictions as it was trained using the results to predict the past. Additionally, using a team's roster information as opposed to those players who received game-time would also assist in making generalizable predictions.

REFERENCES

Miller, R., Schwarz, H., & Talke, I. S. (2017). Forecasting sports popularity: application of time series analysis. *Academic Journal of Interdisciplinary Studies*, *6*(2), 75-75.

Thabtah, F., Zhang, L., & Abdelhamid, N. (2019). NBA game result prediction using feature analysis and machine learning. *Annals of Data Science*, *6*(1), 103-116.

Beckler, M., Wang, H., & Papamichael, M. (2013). Nba oracle. *Zuletzt besuchtam*, *17*(20082009.9).

Curcic, D. (1970, September 03). 70 years of height evolution in the NBA [4,504 players analyzed]. Retrieved December 09, 2021, from https://runrepeat.com/height-evolution-in-the-nba

RF Wireless World. (n.d.). *Advantages of Deep Learning | Disadvantages of Deep Learning*. RF Wireless World. Retrieved December 14, 2021, from https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Deep-Learning.html