Aidan Clark, Ping Hill

DS340

Professor Gold

Introduction

In any professional sports league, players are evaluated by the skills they have displayed thus far, but also by the potential they have to take their talents to new heights. This is no less prevalent in the National Basketball Association, or NBA, where players are given contracts worth hundreds of millions of dollars off of pure expectation to perform at a high level over an extended period. On top of this, billions of dollars are wagered yearly over the performance of these players, and many of these bets are done without data to reinforce their decisions. From the top of a basketball team's front office to everyday sports fans, using data to inform their decision-making in contract negotiation or a simple bet could put them way ahead of the competition.

Motivated by these real-world use cases, this paper sets out to outline the process behind an LSTM model that predicts if an NBA player will regress or improve the following season based on an extensive number of player statistics. The NBA records stat averages for every player including the number of games played, minutes, field goals attempted and made, three-pointers attempted and made, free throws, rebounds, assists, steals, turnovers, blocks, and points. These stats give a general overview of the player's season, with more offensive stats being points, assists, field goal percentage, and defensive-focused stats being blocks and rebounds. A user of this LSTM model can judge a player's projected stats, allowing them to make an informed decision about that player if they wish to bet on them.

Methodology

It was determined that the ideal dataset for this model would be leveraging both an extensive amount of data and staying within the bounds of what many consider the “modern NBA.” For this reason, it was determined that the cutoff point that best encapsulates this era of the NBA while giving a large enough sample of data would be the late 1990s to the early 2020s (2021-2022). This allows the model to have plenty of years to use as training and testing data as well as the opportunity to manually compare predictions made by the model for the 2022-2023 season, where all of the data is readily available.

With this outline in mind, the model was ultimately trained on a dataset created by Eduardo Tocco on the data catalog platform Data World[1]. The data was directly scraped from Basketball-Reference, a reputable source for all NBA-related data[2].

This dataset had 14,573 rows to begin with, which means it contained data upwards of 14,000 season stats of every player who played since the 1997-1998 season. The first issue found was many players included in the dataset had null values on certain features. This mostly occurred in columns where the value was displayed as a percentage, namely FG%, 3PT%, and 2P%. Those who didn’t attempt any of these shots would register their data as NaN, not allowing the model to proceed with training. At first, to account for this error, it was decided it would be best to convert all NaNs into 0. This is an effective method for cleaning the data as the ‘divide by zero’ error causing the null values only occurred when zero shots were taken in a certain category. It was important to address this separately than situations where players had very little playing time since many players, especially in the late 1990s and early 2000s, did not attempt any three-point field goals. That being said, it was later decided to remove the shooting percentages as a whole from the feature list, as they can be calculated by taking the projected

[1] https://data.world/etocco/nba-player-stats
[2] https://colab.research.google.com/drive/1fb4y802jp9b7DE-aEq5aSyQAIhmOqzPF?usp=sharing

made shots divided by the total shots attempted since they are already dependent on those values to begin with.

Another problem with the dataset that needed to be addressed was instances when a player was traded midway through a season. This was an issue as their data would display in multiple rows, the number dependent on the number of teams the player joined throughout a season. Along with these rows, a final row with the team name “TOT” held the individual stat averages for the whole season regardless of team. To address this issue, data entries labeled “TOT” were identified and the surrounding rows with partial season data were removed, ensuring that the model was not trained on partial data, ultimately skewing results.

The third instance of pre-processing involved addressing the large number of players who rarely played. To address this, the model was trained on a dataset that excluded all players who averaged less than 5 points in their career. Though some may disagree with the removal of these players from the training set, it was found that the predictions improved in accuracy for many of the players that receive the majority of the playing time, as the model leaned too heavily on the players with very impact on the game itself, resulting in lower predicted values for players with historically valuable seasons. After these preprocessing steps, the number of rows was reduced to 7,404.

To prepare the data for training in the LSTM model. A list “feature_columns” was created, filled with stats the data should use to train the model. The list ‘x’ was then initialized which held input sequences of the LSTM and list ‘y’ stored the target.

Individual players were looped through, processing each player's data individually, and recording the season to maintain chronological order. For each player, the selected features are converted into a numpy array ‘player_features’, and sequences are made from these features. After each sequence is created, y is set to the next data point.

The sequences are then padded to ensure all inputs are the same length. LSTMs must take sequences of the same length, and problems arise when a player has played 4 seasons

versus 10 seasons. This is done with the value 0.0, and a masking layer is set when creating the model to ignore inputs where all input features are 0.0. This was added to ensure the padding wouldn't influence the predictions.

Why An LSTM?

The model chosen to use was an LSTM because of its ability to take into account time-series and sequence prediction tasks. This is done because LSTMs take in data and learn the temporal dependencies between events in a sequence. This is very important when predicting a player's performance based on historical data. LSTMs also offer flexibility in their model, allowing them to handle different input lengths and allowing padding or masking without introducing bias or interference.

When compared with RNNs, which have problems with a vanishing gradient for long sequences and are less effective at understanding long-term dependencies, LSTMs excel for the type of data in this project. In comparison with Convolutional Neural Networks, CNNs are much better suited for spatial data processing, such as images, and not so much for predictions accounting for temporal order. Compared with Feed-Forward Neural Networks, these do not account for sequences of data points, rather treating each input independently, making it unsuitable for tasks dependent on temporal sequences.

The model chosen was sequential. The first LSTM layer was chosen with 64 units, being able to capture broad representations in the data. The activation was set to ReLu for faster and more efficient training, as well as addressing the vanishing gradient problem. It was determined to set return_sequences = True, so this layer will return the full sequence to the following layer. In the second LSTM layer, it was set to 32 units. This was to condense the information learned from the previous layer and focus on keeping the most important features while preventing overfitting. With return_sequences = False, this layer also only returns a single output vector. Following each LSTM layer, a dropout layer is added with setting fraction (0.2) of the inputs to

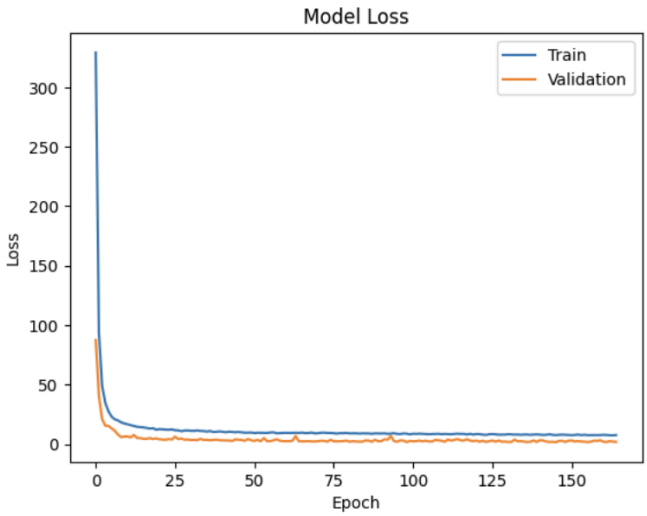
zero to combat overfitting. A dense layer is used to help shape the final output, mapping the output of the LSTM layers to the output shape. To compile the model, an Adam Optimizer was utilized because of its ability to handle vanishing gradients and its efficiency in training deep networks. To test the accuracy of the predictions, Mean Squared Error (MSE) and Mean Absolute Error (MAE) were utilized. MSE is particularly valuable as it emphasizes larger errors, which can be critical in scenarios where accurate predictions of player stats such as points or rebounds have less room for error. MAE provides a straightforward average of the absolute differences in predicted stats, offering clarity on the typical prediction accuracy and ensuring the model's reliability across a variety of performance metrics. Utilizing both MSE and MAE allows for a balanced evaluation of the model. enhancing the predictive accuracy.

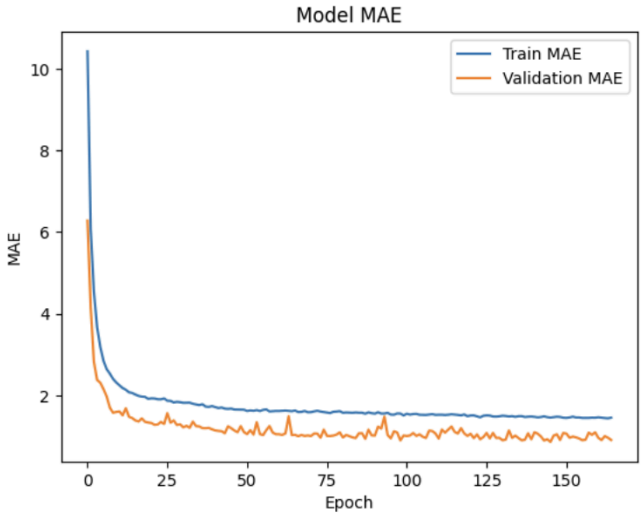
Data Training

To train the data, storage lists were initialized, with X_train, X_val, X_test, and y_train, y_val and y_test, as well as a counter. A loop then iterates over each player's grouped data and counts the number of seasons that specify the player played. Split sizes are calculated with 70% of each player's career data used for training, 15% for validation, and 15% for testing. The data is then extracted from the padded sequences, utilizing the counter and slicing to find the correct segment of data. The total number of seasons, played by this player, which in this case works as an index, is then added to the counter to be able to slice the next player.

The segments of the player's data are then appended to their respective lists utilizing extend, allowing unpacking of the argument and adding them to the list. These lists are then converted to NumPy arrays and in float32 format to be suitable for training.

Accuracy






```
61/61 [==============================] - 1s 12ms/step - loss: 2.5160 - mae: 0.8668
Test Loss: 2.515990972518921, Test MAE: 0.8667515516281128
```

The model had a loss of 2.516, indicating this value as the average squared difference between predicted values and actual values in the validation set. The MAE was 0.8668, showcasing the average absolute difference between predicted and actual values in the validation set. The model produced test losses and MAE that were very similar to the model's validation loss, with a value of 2.5159, and a test MAE of 0.8667.

Coefficient of Determination

```
13/13 [==============================] - 1s 11ms/step
R² score: 0.68
```

The R^2 score, or coefficient of determination measures the amount of variance in the target variable that can be predicted from independent variables.

This is done first by calculating the residual variance, which measures the sum of the squares of the differences between observed values and predicted values. This

$$1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

value is then divided it by the total variance, how much the actual data points deviate from the mean. The resulting value gives an idea of how effective the model is in capturing the underlying relationships and patterns in the data. An R^2 value of 1 means the model's predictions are very close to the actual values, while a score of 0 indicates the model and data fail to explain the variance in the predictions. In this case, a value of 0.68 indicates that 68% of the variance can be explained by the independent variables in the data.

Mean Absolute Percentage Error

```
13/13 [==============================] - 0s 16ms/step
Mean Absolute Percentage Error (MAPE): 30.00%
```

The Mean Absolute Percentage Error is used to measure the accuracy of predictions as a percentage. This is done by first calculating the difference between actual values and the predicted values, and then dividing by the actual values. The absolute value is then taken to ensure the value is not negative. These errors are summed up, and then averaged over total observations and multiplied by 100, providing the magnitude of error in the form of a percentage. In the case of this model, the MAPE value was 30%, indicating that on average, the predictions were 30% off the actual values. Given the high variance of NBA statistics, these numbers were better than expected.

$$\frac{1}{N}\sum_{t=1}^{N}\left|\frac{E_t - A_t}{A_t}\right|$$

Results

The model was tested on many players in different stages of their careers. One that exemplifies the model's prediction accuracy is Kyle Lowry. Given that Kyle Lowry entered the league in 2006, the model was provided with plenty of data to base its predictions on. Below are the 2022-23 season predictions for Kyle Lowry and his actual statistics for the regular season (highlighted in orange).

```
G                       63.849
GS                      62.799
MP                      33.054
FG                      4.419
FGA                     10.834
3P                      1.530
3PA                     4.267
2P                      2.907
2PA                     6.544
ORB                     0.769
DRB                     3.280
TRB                     4.029
AST                     4.122
STL                     1.202
BLK                     0.248
TOV                     1.898
PF                      2.170
PTS                     12.856
FG%                     0.408
3P%                     0.359
2P%                     0.444
```

Season	Age	Tm	Lg	Pos	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	eFG%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
2006-07	20	MEM	NBA	PG	10	0	17.5	1.4	3.8	.368	0.3	0.8	.375	1.1	3.0	.367	.408	2.5	2.8	.893	1.2	1.9	3.1	3.2	1.4	0.1	1.2	2.0	5.6
2007-08	21	MEM	NBA	PG	**82**	9	25.5	3.1	7.2	.432	0.4	1.7	.257	2.7	5.5	.487	.463	3.0	4.2	.698	0.5	2.5	3.0	3.6	1.1	0.3	1.5	2.3	9.6
2008-09	22	TOT	NBA	PG	77	21	21.8	2.5	5.6	.435	0.3	1.3	.255	2.1	4.4	.488	.464	2.4	2.9	.801	0.3	2.1	2.5	3.6	0.9	0.2	1.5	1.9	7.6
2008-09	22	MEM	NBA	PG	49	21	21.9	2.3	5.6	.412	0.3	1.4	.246	1.9	4.1	.468	.443	2.6	3.3	.801	0.3	2.0	2.3	3.6	1.0	0.2	1.6	1.8	7.6
2008-09	22	HOU	NBA	PG	28	0	21.7	2.8	5.8	.475	0.3	1.0	.276	2.5	4.8	.519	.500	1.9	2.3	.800	0.5	2.3	2.8	3.5	0.8	0.3	1.4	2.0	7.6
2009-10	23	HOU	NBA	PG	68	0	24.3	2.7	6.8	.397	0.5	2.0	.272	2.2	4.8	.448	.436	3.2	3.8	.827	1.3	2.3	3.6	4.5	0.9	0.1	1.7	2.5	9.1
2010-11	24	HOU	NBA	PG	75	71	34.2	4.6	10.8	.426	1.7	4.6	.376	2.9	6.2	.464	.506	2.6	3.3	.765	1.2	2.9	4.1	6.7	1.4	0.3	2.1	2.8	13.5
2011-12	25	HOU	NBA	PG	47	38	32.1	4.5	10.9	.409	1.7	4.5	.374	2.8	6.4	.434	.486	3.6	4.2	.864	0.8	3.7	4.5	6.6	1.6	0.3	2.8	2.8	14.3
2012-13	26	TOR	NBA	PG	68	52	29.7	3.7	9.2	.401	1.5	4.1	.362	2.2	5.1	.433	.482	2.8	3.5	.795	0.8	3.9	4.7	6.4	1.4	0.4	2.3	3.2	11.6
2013-14	27	TOR	NBA	PG	79	79	36.2	5.8	13.7	.423	2.4	6.3	.380	3.4	7.3	.460	.511	4.0	4.9	.813	1.1	3.6	4.7	7.4	1.5	0.2	2.5	3.4	17.9
2014-15 ★	28	TOR	NBA	PG	70	70	34.5	6.1	14.9	.412	1.9	5.6	.338	4.3	9.3	.457	.476	3.6	4.5	.808	0.8	3.9	4.7	6.8	1.6	0.2	2.5	3.0	17.8
2015-16 ★	29	TOR	NBA	PG	77	77	37.0	6.6	15.6	.427	2.8	7.1	.388	3.9	8.5	.461	.516	5.2	6.4	.811	0.7	4.0	4.7	6.4	2.1	0.4	2.9	2.7	21.2
2016-17 ★	30	TOR	NBA	PG	60	60	37.4	7.1	15.3	.464	3.2	7.8	.412	3.9	7.5	.518	.569	5.0	6.1	.819	0.8	4.0	4.8	7.0	1.5	0.3	2.9	2.8	22.4
2017-18 ★	31	TOR	NBA	PG	78	78	32.2	5.2	12.1	.427	3.1	7.6	.399	2.1	4.5	.474	.553	2.9	3.3	.854	0.8	4.7	5.6	6.9	1.1	0.2	2.3	2.5	16.2
2018-19 ★	32	TOR	NBA	PG	65	65	34.0	4.7	11.4	.411	2.4	7.0	.347	2.3	4.4	.514	.518	2.5	3.0	.830	0.6	4.2	4.8	8.7	1.4	0.5	2.8	2.6	14.2
2019-20 ★	33	TOR	NBA	PG	58	58	36.2	5.8	13.8	.416	2.8	8.0	.352	2.9	5.8	.504	.518	5.1	5.9	.857	0.6	4.5	5.0	7.5	1.4	0.4	3.1	3.3	19.4
2020-21	34	TOR	NBA	PG	46	46	34.8	5.7	13.0	.436	2.8	7.2	.396	2.8	5.8	.487	.546	3.0	3.5	.875	0.8	4.6	5.4	7.3	1.0	0.3	2.7	3.1	17.2
2021-22	35	MIA	NBA	PG	63	63	33.9	4.4	10.0	.440	2.3	6.1	.377	2.1	3.9	.539	.555	2.3	2.8	.851	0.5	4.0	4.5	7.5	1.1	0.3	2.7	2.8	13.4
2022-23	36	MIA	NBA	PG	55	44	31.2	3.6	8.8	.404	1.9	5.6	.345	1.6	3.2	.509	.514	2.1	2.5	.859	0.8	3.3	4.1	5.1	1.0	0.4	1.9	2.6	11.2

3

Notable predicted statistics include 12.85 points, a field goal percentage of 0.408, and 1.898 TOV (turnovers). Compared to his actual 2022-2023 season stats, these predictions are almost identical, with Lowry averaging 11.2 points, a FG% of 0.404, and 1.9 TOV. Given Lowry's position as Point Guard, minimizing turnovers is one of his most important tasks, especially when adjusting to a new team (he was traded to the Miami Heat the year prior). Though the point total was slightly off, the model successfully predicted Lowry would reduce his scoring, an expectation that is standard as a player approaches the late stage of his career. Generally, the

[3] https://www.basketball-reference.com/players/l/lowryky01.html

more seasons the model could analyze for a player, the better it performed in terms of accuracy. This was an expectation given players in the early stages of their career are still adjusting to the physicality and skill level of the NBA and explosions of efficiency and scoring are far more common in the early-mid section of one's career as opposed to the late end of a player's professional career.

Conclusion

While this was an interesting subject to analyze, statistics are just one part of the puzzle that makes up a player's career expectations. These basic stats can tell a lot about a player's specialties and how they stack up against the rest of the NBA, but a field goal percentage or blocks per game will not tell you how much effort the player puts in at practice, the chemistry between him and his teammates, and the unfortunate but very real possibility that the player may suffer a season-ending injury with one bad landing. Our model was able to predict a lot of the ups and downs of particular players and it was not afraid to make bold predictions about star player downturns following historic seasons, but it could be just as wrong as it was right due to the high variance that exists in professional basketball.

The stages of development were typical of any ML project. Initially, we began with plans of predicting solely Points Per Game (PPG) for star players. Ultimately, this seemed too small a scale of a project with limited use cases outside of an interesting thought experiment (though betting on PPG is certainly still possible). As a result, we decided to train on multiple major statistics that define NBA players' talents. We kept pretty close to our initial plans to use an LSTM due to the nature of our data (time-series, sequential). Throughout the process of this project, we made many changes that were described earlier in the paper, namely the decision to omit certain seasons and features since they were skewing the predictions in ways that led to less accurate results. In doing so, we became familiar with the process of training and curating model inputs to create the best model for our use case. Given our shared interest in the NBA,

we spent a lot of time on this project, making time to work together with an even split of the workload.

Future Considerations

As aspiring ML engineers, this experience was as informative as it was humbling to learn that tweaking a model with so many variables can be such a daunting task. More time and care would be required for a model like this to be profitable for a sports bettor in the long term, even more so for an NBA team's general manager, where picking the wrong player in the draft or free agency can cost them billions. Given more time, resources, and computing power, this model could have a tremendous impact on closing the gap of data-driven decision-making between the bettors that always seem to have the winning wager, and those that consistently turn in losses in the long term, but adjustments would certainly need to be made to account for all the other variables involved in the game, some not even showing up on a stat sheet.