Predicting MLB Runs Scored via Neural Networks

Miguel Corona

Dec 2020

Abstract

Major League Baseball is deemed unfair due to the league's lack of enforcing a salary cap that places a disadvantage on small market teams. Wealthy teams are capable of signing the best players available on the market due to their financial resources. The discrepancy in wealth distribution led to the development of the Moneyball concept pioneered by the 2002 Oakland Athletics as a means to bridge the competitive gap. Moneyball relies on the analysis of players via data instead of assessing players according to the traditional baseball ideals. The analysis of data provides insight into the team's potential production based on team statistics instead of individual players.

The objective of this paper is to document the efforts of using neural networks to recreate the conclusions of the Oakland A's that has influenced MLB roster composition since Moneyball's inception. Neural networks were selected due to their ability to learn underlying properties of the data and provide reasonable predictions such as predicting a team's production. The models were constructed and trained with the team's offensive metrics across multiple MLB seasons to determine a team's running scoring potential within a season regardless of the players within the organization and the organization's payroll. Although the means of how the game has changed over time, the games are ultimately won by the team that scores the most runs. The number of wins can then be approximated by the number of runs a team scores throughout the season. The team's wins are not tied directly to any one player and Moneyball reinforces the thought that the sum of the parts outweighs any individual player.

1 Introduction

Baseball is a sport in which the winner is determined by the team that has more runs at or after nine innings of play. The team with the more talented roster is deemed the favorite and the quality of the roster may be influenced by each team's financial resources. Wealthier teams benefit from the lack of a

salary cap as it permits them to use their capital to pursue talented players with offers that cannot be matched by small market teams. Moneyball is a means for small market teams to compete against wealthier opponents by recreating the best players' output as an aggregate of multiple, less expensive options via understanding the data as opposed to the players. Peter Brand, a character from the film Moneyball, provided his insight, "Your goal shouldn't be to buy players. Your goal should be to buy wins. In order to buy wins, you need to buy runs"[1].

The correlation between Runs Scored and wins was proposed by baseball statistician Bill James and is depicted in Figure 1. A linear relationship exists between the actual wins and the wins projected by the Pythagorean approximation. The Pythagorean Theorem of Baseball[3] provides an approximation of wins based on the Runs Scored and Runs Allowed calculated via

$$W\% = \frac{RS^2}{RS^2 + RA^2} \tag{1}$$

A neural network shall be constructed capable of predicting Runs Scored based on the statistics: Batting Average which measures the probability of a hit per at-bat, On-Base Percentage which measures the probability a batter becomes a runner, and Slugging Percentage which measures the expected bases per at-bat.

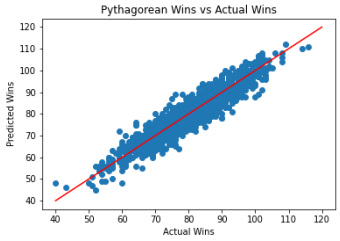


Figure 1: Pythagorean Approximation

2 Data Analysis

2.1 Dataset

The dataset was acquired from Kaggle as the "Moneyball Dataset"[2]. It contains the statistics of interest for all MLB teams from the 1969 season through the end of the 2012 season for a total of 1232 entries. The statistics within the dataset are: Runs Scored (RS), Runs Allowed (RA), Wins, On-Base Percentage (OBP), Slugging Percentage (SLG), Batting Average (BA), and Playoffs. A subset of the features shall be utilized for analysis.

2.2 Input: BA, OBP, SLG

The statistics of interest for analysis are OBP, SLG, and BA. The distribution of the input data are depicted in Figure 2 and an analysis of the data is contained within Table 1. The statistics of interest are centered about a mean and tail off towards the ends simulating a bell-like distribution. The data is void of any apparent biases based on the distribution of the inputs.

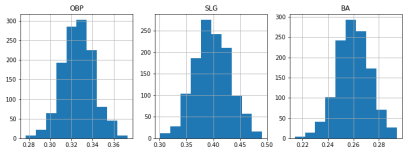


Figure 2: Input Data Distribution

Metric	Mean	Std	Min	Max
OBP	0.326	0.015	0.277	0.373
SLG	0.397	0.0333	0.301	0.491
BA	0.259	0.0129	0.214	0.294

Table 1: Input Data Statistics

2.3 Output: Runs Scored

The neural network shall aim to predict is the Runs Scored. The distribution of the output is depicted in Figure 3 and an analysis of the data is contained within Table 2. The output appears to be void of any apparent biases and has a bell-like distribution centered about a mean.

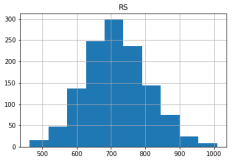


Figure 3: Output Data Distribution

2.4 Initial Assessment

An initial assessment was performed by observing the correlation between the desired inputs and outputs. The heat map is depicted in Figure!4 and indicates

Metric	Mean	Std	Min	Max
RS	715.082	91.534	463	1009

Table 2: Output Data Statistics

that there is a stronger correlation between the Runs Scored and the metrics OBP and SLG than with BA.

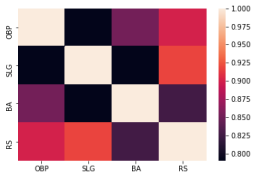


Figure 4: Input/Output Correlation

2.5 Data Normalization

The input data was normalized through a Z-Score calculation for each element x within X. The Z-Score calculation is performed via

$$\frac{x - \mu}{\sigma} \tag{2}$$

The rationale for selecting Z-Score for normalization was to minimize deviations in which the game was played such as the offensive surge during the steroid era. The steroid era is a blurry timeline where the league failed penalize players using performance enhancing drugs that resulted in questionable offensive metrics. The surge of offensive metrics may make other normalization techniques such as max-mean normalization not viable.

3 Building a Model

3.1 Data Split

The development set was partitioned such that the training set is composed of 75% of samples and the remaining 25% are reserved for the validation set. The split was performed through the usage of seasons so that each set contains low scoring and high scoring teams.

3.2 Linear Regression Model

A linear regression model was developed to establish a baseline for the neural networks as the objective is to construct a model capable of exceeding linear

regression. The linear regression model was trained on the training set and its performance was evaluated on the validation set. The learning curves of the linear regression model are depicted in Figure 5 and indicate that the model lacks the capacity to learn meaningful information as no convergence of loss occurs. The loss metrics contained within Table 3 further demonstrate that the model performs poorly due to the model's limitation.

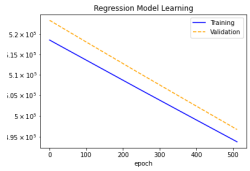


Figure 5: Linear Regression Model Learning Curve

Set	MSE	MAE
Train	493 790.260	698.981
Valid	493 736.409	700.851

Table 3: Linear Regression Loss Metrics

3.3 Candidate Models

A number of candidate models were developed and evaluated to identify a suitable network for predicting the Runs Scored. The architectures of these candidates are contained within Figure 6. Each model was trained on the training set and evaluated on the validation set. The best network shall be the one that yields the minimal loss, MAE, on the validation set as the validation loss serves as an estimate for the generalization loss.

The resulting learning curves of the candidate models are depicted in Figure 7. The learning curves indicate that each model is capable of learning from the data albeit with varying rates. Model 2 has the simplest architecture yet it manages to yield a comparable loss between the training and validation sets much more slowly than the other candidates. The impact of network depth is depicted in the learning curves of Model 1 and Model 2. Model 1 converges quicker than Model 2 even though the models have same number of neurons with the distinction that Model 1 has the neurons split across two hidden layers instead of having them in one. Model 5 is an augmented version of Model 1 with an additional layer of 24 neurons and its loss converges the quickest though it suffers from more pronounced fluctuations in the validation loss. The number of neurons also affects the convergence as demonstrated by the learning curves of Model 3 and Model 4. The learning curve of Model 3 is similar to that of Model 1

Model: "sequential_13"

Layer (type)	Output Shape	Param #
dense_51 (Dense)	(None, 18)	72
dense_52 (Dense)	(None, 6)	114
dense_53 (Dense)	(None, 3)	21
dense_54 (Dense)	(None, 1)	4

Total params: 211
Trainable params: 211
Non-trainable params: 0

(a) Model 1

Model: "sequential_14"

Layer (type)	Output Shape	Param #
dense_55 (Dense)	(None, 18)	72
dense_56 (Dense)	(None, 9)	171
dense_57 (Dense)	(None, 1)	10

Total params: 253
Trainable params: 253
Non-trainable params: 0

(b) Model 2

Model: "sequential_15"

Layer (type)	Output Shape	Param #
dense_58 (Dense)	(None, 36)	144
dense_59 (Dense)	(None, 12)	444
dense_60 (Dense)	(None, 1)	13

Total params: 601
Trainable params: 601
Non-trainable params: 0

(c) Model 3

Model: "sequential_16"

Layer (type)	Output Shape	Param #
dense_61 (Dense)	(None, 18)	72
dense_62 (Dense)	(None, 12)	228
dense_63 (Dense)	(None, 6)	78
dense_64 (Dense)	(None, 1)	7

Total params: 385
Trainable params: 385
Non-trainable params: 0

(d) Model 4

Model: "sequential_17"

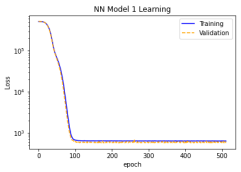
Layer (type)	Output Shape	Param #
dense_65 (Dense)	(None, 24)	96
dense_66 (Dense)	(None, 18)	450
dense_67 (Dense)	(None, 6)	114
dense_68 (Dense)	(None, 3)	21
dense_69 (Dense)	(None, 1)	4

Total params: 685
Trainable params: 685
Non-trainable params: 0

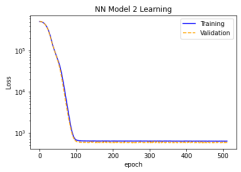
(e) Model 5

Figure 6: Candidate Neural Networks

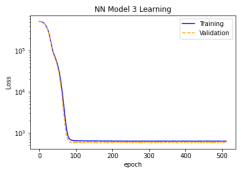
although Model 3 is a variant of the shallower Model 2 with additional neurons. The convergence rate of Model 4 was quicker than Model 1 as it is a variant of Model 1 with the same depth and additional neurons.



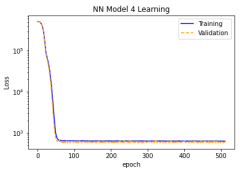
(a) Model 1



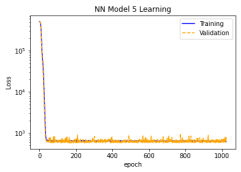
(b) Model 2



(c) Model 3



(d) Model 4



(e) Model 5

Figure 7: Candidate Models' Learning Curves

The loss metrics are contained within Table 4. Each candidate exceeds the performance of the linear regression model based on the validation loss, MAE. The linear regression model is a simple model that lacks complexity to learn due it to having four trainable parameters compared to the hundreds for each candidate. The loss metrics demonstrate the effects of the model's trainable parameters determined by the hyperparameters. The effects of network depth are demonstrated in the metrics of Models 1, 2, and 5. Model 1 outperforms the shallower Model 2 and the deeper Model 5 based on the validation loss

suggesting an appropriate depth is required. Failing to meet or exceeding the depth requirement fails to produce an optimal model. The impact of the model's neurons is demonstrated via the comparisons between Model 1 with Model 4 and Model 2 with Model 3. The simpler Model 1 and Model 2 outperformed their more complex counterparts in terms of validation loss. Complex models yield more reliable results than the linear regression model, but cannot generalize the data as well as simpler models.

Set	MSE	MAE
Train	634.176	19.943
Valid	578.203	19.379

(a) Model 1

Set	MSE	MAE
Train	629.532	19.809
Valid	592.288	19.616

(b) Model 2

Set	MSE	MAE
Train	629.670	19.983
Valid	589.404	19.727

(c) Model 3

Set	MSE	MAE
Train	658.842	20.497
Valid	621.380	20.405

(d) Model 4

Set	MSE	MAE
Train	812.190	22.307
Valid	810.396	22.287

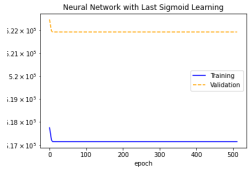
(e) Model 5

Table 4: Candidate Models Loss Metrics

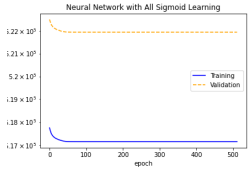
3.4 Effects of Incorrect Activation Functions

The effects of utilizing incorrect activation functions were observed on the proposed model. Predicting the Runs Scored is a regression problem and requires a linear activation function for the last layer. Two additional training cycles were performed on the selected candidate. One training cycle involved setting the last layer to use the sigmoid activation function and the second involves setting multiple layers to have a sigmoid activation function.

The learning curves of both experiments are contained within Figure 8. The learning curves indicate that the model is unable to yield a meaningful output if there is at least one incorrect activation function within the network. The model's ability to learn is dependent on the correct usage of activation functions.



(a) Last Layer Sigmoid



(b) All Layers Sigmoid

Figure 8: Effects of Incorrect Activation Functions

4 Overfitting Models

Overfitting is the phenomena in which the model fails to generalize data and only performs well on the training set. The training loss and validation loss diverge such that the model can only provide reliable predictions for the training data. Factors such as an overly complex architecture or prolonged training can contribute to overfitting. This portion of the experiment documents the efforts involved in identifying overfitting models.

4.1 Identifying an Overfitting Model

A model whose architecture is more complex than the underlying patterns within the data is capable of overfitting. Overly complex networks are capable of memorizing the output of each training sample and may fail to provide a reasonable output for unseen data. One model identified to be capable of overfitting is depicted within Figure 9a. The overfitting model is more complicated than the candidate models in terms of the depth and the number of neurons per layer. The model's training requires an increase of epochs to account for the additional model complexity.

Model: "sequential_8"

Layer (type)	Output Shape	Param #
dense_28 (Dense)	(None, 256)	1024
dense_29 (Dense)	(None, 128)	32896
dense_30 (Dense)	(None, 64)	8256
dense_31 (Dense)	(None, 32)	2080
dense_32 (Dense)	(None, 16)	528
dense_33 (Dense)	(None, 8)	136
dense_34 (Dense)	(None, 1)	9

Total params: 44,929
Trainable params: 44,929
Non-trainable params: 0

(a) Overfit

Model: "sequential_4"

Layer (type)	Output Shape	Param #
dense_20 (Dense)	(None, 162)	810
dense_21 (Dense)	(None, 81)	13203
dense_22 (Dense)	(None, 9)	738
dense_23 (Dense)	(None, 3)	30
dense_24 (Dense)	(None, 1)	4

Total params: 14,785
Trainable params: 14,785
Non-trainable params: 0

(b) Overfit with Output as Input

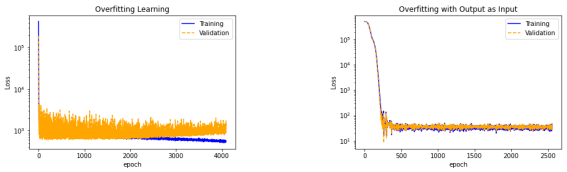
Figure 9: Overfitting Models

The learning curves of the overfitted model are depicted in Figure 10a. The overfitted model yields a learning curve whose validation loss slightly diverges from the training loss after numerous epochs. The loss metrics contained within

Table 5a yield noticeable differences between the training and validation losses. The model requires additional training to yield a comparable training loss to the candidate models, but there's no indication that the validation loss will improve due to the observed divergence in loss. The simpler models are better suited to generalize the data and converge much quicker than complex models.

4.2 Overfitting with Output as an Input

A simpler architecture capable of overfitting can be identified by including the each sample's output as a feature. The training set was modified to include the output an additional feature and the values were normalized via the Z-Score. The normalization was only applied to the input version of the feature and not the actual output. The proposed model believed to be capable of overfitting provided that the input includes its output is depicted in Figure 9b. The training requires an increase in epochs to account for the added model complexity and an increase in batch size to alleviate the increased susceptibility due training with output as a feature.



(a) Overfit

(b) Overfit with Output as Input

Figure 10: Overfitting Models Learning Curves

Set	MSE	MAE
Train	720.404	21.780
Valid	1096.954	26.651

(a) Overfitting Model

Set	MSE	MAE
Train	25.221	4.303
Valid	26.150	4.366

(b) Overfitting Model Output as Input

Table 5: Overfitting Models Loss Metrics

The learning curves of the overfitted model with the output as an input are depicted in Figure 10b. The validation loss does not appear to diverge and aligns with the training loss. The model may require additional epochs to potentially diverge as observed in the more complex overfitting model. The model's loss metrics are contained within Table 6b and are the minimum training and validation losses observed. An increase in the differences between the training loss

and validation loss may be observed once the losses diverge through additional epochs.

5 Feature Significance and Reduction

Although the models were trained according to the available features, each feature's impact in the model's learning can vary. The proposed model capable of predicting the Runs Scored given the all inputs can also identity the significance of each feature. The model can be potentially improved by identifying the significant features and training with a reduced set composed of the meaningful features.

5.1 Feature Significance

The significance of a feature can be determined by evaluating a model's performance on the validation set after the model is trained with only the feature of interest. The resulting models are not expected to outperform the preceding models, but to identify each feature's relative impact. The training and assessments involve using tensor input consisting of only the feature of interest.

The resulting learning curves for each single feature training are contained within Figure 11. The training loss and validation loss converge indicating that the model can learn with a single feature. The validation loss metrics are contained within Table 6a and represented within Figure 12a. The loss metrics indicate that Batting Average is not a reliable statistic to assess a team's run scoring potential as it yields the greatest loss. Slugging Percentage and On-Base Percentage are strong indicators for a team's runs scoring potential as these yield comparable losses. The significance of each feature aligns with the correlations observed in the initial assessment.

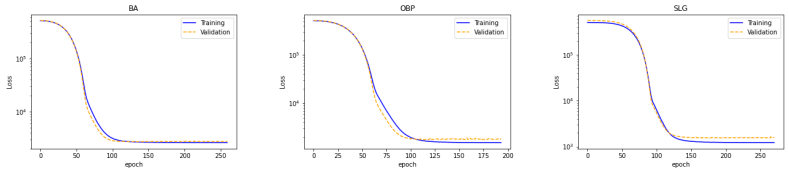
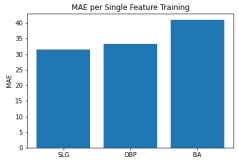


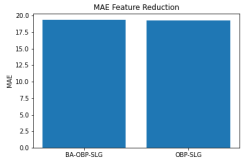
Figure 11: Single Feature Learning Curves

Interestingly, the difference between Batting Average and On-Base Percentage is much greater than the difference between On-Base Percentage and Slugging Percentage and this may be due information conveyed by the statistic. Batting Average may be a poor indicator for Runs Scored as it only provides the probability of a hit occurring during the at-bat and treats all hits equally. A homerun and a single are the same in the context of Batting Average though each yields a

different outcome. The importance of the type of hit is exemplified by training observed with Slugging Percentage as the statistic weighs the hits to provide an expectation value and the statistic was determined to be the most significant. The On-Base Percentage also serves as a strong indicator for Runs Scored even though it also treats all hits equally. The reasoning for On-Base Percentage being a reliable statistic may be due to it accounting for other means of the batter becoming a runner. A runner must be on base to potentially score.



(a) Single Feature Validation Loss



(b) Feature Reduction Validation Loss

Figure 12: Feature Significance and Reduction Loss

Feature	MAE
SLG	29.489
OBP	33.141
BA	40.602

(a) Single Feature Validation Loss

Features	MAE
BA-OBP-SLG	19.370
OBP-SLG	19.270

(b) Feature Reduction Validation Loss

Table 6: Feature Significance and Reduction Loss Metrics

5.2 Feature Reduction

The objective of the feature reduction experiment was to observe the model's performance by omitting the least significant features in the model's training. Only a single feature was omitted as additional omissions replicate the findings of the preceding subsection.

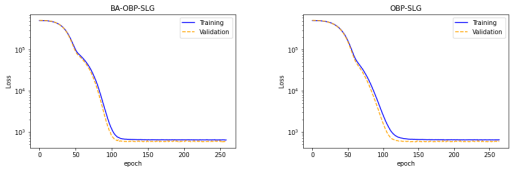


Figure 13: Feature Reduction Learning Curves

The resulting learning curves for feature reduction training are contained within Figure 13b. The learning curves indicate that the model generalized the data as the validation loss aligns with the training loss regardless of the inputs. No overfitting was observed with the reduced input even though the hyperparameters were optimized to account for all features. Training terminates much sooner for the reduced input model and this can be attributed to the reduction of trainable parameters due to the omission of features. The observed validation losses for the feature reduction experiment are contained within Table 6 and depicted within Figure 12b respectively. The omission of the Batting Average minimally affects the model's performance as training with and without the statistic yield comparable validation losses. The On-Base Percentage and Slugging Percentage provide sufficient information to reliable predict the Runs Scored.

6 Conclusion

The objective was met as a neural network capable of predicting a team's Runs Scored based on Batting Average, On-Base Percentage, and Slugging Percentage was developed. The model's predictions are not exact, but provide ballpark estimates that are more reliable than those of a linear regression model. The process of identifying a suitable model consisted of constructing candidate models, training each model, and evaluating each model with the validation set. The validation set was pivotal in understanding whether the model was generalizing the data or potentially overfitting. Each candidate was capable of minimizing the training loss, but complex models do not necessarily yield the minimal validation loss. The hyperparameters such as the network depth and number of neurons were adjusted based on the validation loss to yield a model capable of generalizing.

The model development process not only identified the appropriate architecture, but it also revealed the features to emphasize for predicting a team's output. Each feature's significance was determined by training the model with the statistic of interest and evaluating the resulting model's performance on the validation set. Slugging Percentage and On-Base Percentage were identified as the significant metrics to assess a team's run scoring potential. Batting Average which held a considerable weight in evaluating player performance was determined to be a flawed statistic to measure a team's output. The statistic has negligible benefit in the model's learning as omitting the statistic yields a model whose performance is comparable to one that includes it. The model emphasizes Slugging Percentage which differs from the Moneyball founders' emphasis on On-Base Percentage. The deviation in emphasis may be due to the variability in Slugging Percentage as the statistic accounts for at-bats as opposed to plate appearances and these are distinct. Future considerations for predicting Runs Scored may involve the usage of the On-Base Plus Slugging statistic which accounts both On-Base Percentage and Slugging Percentage.

References

[1] Moneyball. Dir. Bennett Miller. Columbia Pictures, 2011. Film

[2] WesDuckett. Moneyball Dataset, 2017

[3] "Pythagorean Theorem of Baseball", Baseball Reference, Sports Reference LLC, https://www.baseball-reference.com/bullpen/Pythagorean_Theorem_of_Baseball