Authors

Victor Verma

Easwer Raman

Introduction

The problem we're trying to solve is predicting offensive statistics of football players so that we can better choose what players to start on fantasy football and what bets are most profitable. We wanted to work on this because we both have a passion for football and have watched it consistently for many years. This made it interesting because we can compare our own predictions on a player's performance with the algorithm's prediction. Many of the sportsbooks also have their own algorithm to determine player stats as well. We wanted to include offensive player performance with stats such as passing yards, rushing yards, completions, etc to be used to compare to betting lines. We were able to implement this algorithm for quarterbacks, running backs, wide receivers, and tight ends. Passing, rushing, and receiving yards are less variable and refer to how many yards a quarterback has thrown the ball for, a running back has run with the ball for, and a wide receiver, tight end, or running back have gotten after they catch and run with the ball. Receptions refer to the number of catches a running back, wide receiver, or tight end and when a quarterback throws a pass and it's caught it is a completion. Every time the quarterback throws the ball or a running back gets handed the ball to run it is an attempt. Lastly, touchdowns refer to when an offensive player throws, runs, or catches a ball in the end zone, while interceptions is when the quarterback throws it and it is caught by a defensive player. One of the major differences between our algorithm and a sportsbooks algorithm is that the sportsbook takes weather, defensive coverages, and other features that we didn't get to implement because we don't really know the best way to implement them due to

either no real formulas to implement it or little access to data. We know how these would affect players' performances qualitatively, but quantifying it is difficult.

Methodology

Due to the time series nature of the data, the Long Short-Term Memory (LSTM) and Simple Recurrent Neural Network (Simple RNN) architectures were chosen. For both architectures, the training data was a 2-D array, with each row representing the player's statistics in that week, depending on the position. For all of the models, mean squared error was used as the loss function, adam was used as the optimizer, and 4 times steps were used. To ensure that the model always has enough training data for the time steps, a player must have at least 6 weeks of statistics available to be used in the model.

For quarterbacks, the features used were completions, attempts, passing yards, passing touchdowns, interceptions, passer rating, sacks, rushing attempts, rushing yards, and rushing touchdowns. For running backs, the features used were rushing attempts, rushing yards, rushing touchdowns, targets, receiving yards, and snap percentage. For wide receivers and tight ends, the features used were targets, receptions, receiving yards, receiving touchdowns, and snap percentage. After extensive testing, it was determined that the best parameters for the QB LSTM were 40 epochs and a batch size of 4, while the best parameters for the QB Simple RNN were 20 epochs and a batch size of 4. For the RB, WR, and TE LSTM, the ideal parameters were 20 epochs and a batch size of 4, and for the RB, WR, and TE Simple RNN, the best parameters were 8 epochs and a batch size of 4.

We used data from pro football reference to implement defensive statistics and the influence that a defense can have on an offensive player's stats. First, we downloaded the data as an Excel workbook and processed it in Excel in order to get the proportion of difference a

defense performs in comparison to the average defense. For quarterbacks, we had average differences in completions, attempts, yards, touchdown percentage, interception percentage, and rating. For running backs, we had average differences in attempts, yards, touchdowns, and yards per game. These were used to multiply the different features that the models predicted in order to get the most optimized prediction of stats. Wide receiver and tight end stats were not predicted because they are coverage and individual defensive player-based rather than being based on the whole defense or defensive line like quarterbacks and running backs respectively.

Results

The models were evaluated in two main ways. First, we looked at both the loss and the validation loss. The goal was to minimize both but avoid overfitting the model. This was a difficult balancing act, mainly due to the lack of training data. Since each model is trained on the weekly statistics of a singular player, if we ran it for the current NFL week (14), then there were usually only 12 rows of training data. The QB model used 10 features, so we were able to train 40 epochs, but the RB, WR, and TE models only used 6, 5, and 5 features, respectively. This is why we could only train those for 8 or 20 epochs depending on the model. The small size of the dataset combined with the small number of features made it very easy to overfit.

The second method of evaluation was comparing the predicted model output to the actual stats a player had in that week. For example, Patrick Mahomes had the following statistics for his game in week 13 against the Packers:

	Completions	Attempts	Pass Yards	Pass TD	Interceptions	Rating	Sacks	Rush Attempts	Rush Yards	Rush TD
LSTM	22.4	33.3	218.4	1.32	0.47	89.4	1.6	4.2	19.0	0.0
Simple RNN	22.9	34.6	239.2	1.97	0.1	89.4	1.6	3	8.7	0.0
Actual	21	33	210	1	1	79.1	3	4	26	0

In general, we noticed that the model was better for statistics that have a greater range like completions, attempts, and passing yards, and less ideal for statistics that have a smaller range like passing touchdowns and interceptions. This generality remained true for running backs, wide receivers, and tight ends as well. From a football perspective, we think that this phenomenon makes sense. In any given NFL game, it is likely that just due to the way the game is played, quarterbacks will tend to compile the statistics with more range no matter what. However, smaller-scale statistics like touchdowns and interceptions are a bit rare, and also a bit more flukey, so it is not as "guaranteed" that a quarterback will produce the expected touchdowns and interceptions. The same is true for running backs, wide receivers, and tight ends.

Considering the smaller scale percentages through the lens of percent error, it should also be noted that a small variation, like 2 touchdowns when only 1 was predicted is a much greater error (50%) than a large variation like 50 extra yards when 300 was predicted (17%). This makes it even more logical that smaller-scale statistics are harder to predict because small errors in terms of absolute size tend to be larger errors in relative size.

Conclusions

In conclusion, we found that the algorithm can predict football player stats with a success rate of 66.7% and 74.4% against the sportsbooks, counting and not counting injuries respectively. Moreover, statistics that had a larger range like passing, rushing, and receiving yards were easier to predict compared to statistics like touchdowns and interceptions, as they are rarer and have a lower relative frequency. We also concluded that this algorithm produces projections based on the history of a player's stats and the defense they are playing, but not based on things like game scripts or injuries that a human would consider. However, our final accuracy stats when compared to a sportsbook are incredible, as the sportsbook wins against the most people. Therefore, we hit our goal of writing an algorithm that can beat the sportsbooks. Lastly, an algorithm like this might be better built for basketball because there are more games in a season, and there are a significantly larger number of advanced stats available to the public. Therefore, it would be easier to implement a more complicated algorithm and the algorithm would have more training data.

Accuracy Data Tables

Quarterback Model Statistics

Base Parameters						LSTM				
Player	**Season**	**End Training Week**	**Time Steps**	**Loss Function**	**Optimizer**	**Layers**	**Epochs**	**Batch Size**	**Loss**	**Validation Loss**
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	10	4	0.1522	0.1036
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	10	6	0.1788	0.1399
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	10	8	0.2171	0.1306
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	25	4	0.1053	0.0896
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	25	6	0.1184	0.0749
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	25	8	0.1162	0.0823
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	40	4	0.079	0.0744
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	40	6	0.0772	0.0799
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	40	8	0.1185	0.0766
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2)	40	4	0.0612	0.0714
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2)	40	6	0.0838	0.0664
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2)	40	8	0.087	0.0714
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2) LSTM (256 units) Dropout (0.2)	40	4	0.0644	0.0675
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2) LSTM (256 units) Dropout (0.2)	40	6	0.0865	0.0671
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	40	8	0.0832	0.0715

						LSTM (128 units) Dropout (0.2) LSTM (256 units) Dropout (0.2)				

Base Parameters						RNN				
Player	**Season**	**End Training Week**	**Time Steps**	**Loss Function**	**Optimizer**	**Layers**	**Epochs**	**Batch Size**	**Loss**	**Validation Loss**
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	10	4	0.0952	0.0717
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	10	6	0.0861	0.0853
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	10	8	0.0884	0.0782
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	20	4	0.0461	0.0883
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	20	6	0.0438	0.0932
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	20	8	0.064	0.0835
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	25	4	0.0245	0.0633
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	25	6	0.0272	0.0925
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	25	8	0.0571	0.0749
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	40	4	0.0167	0.0791
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	40	6	0.0201	0.0675
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	40	8	0.0137	0.081
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(1, "linear")	20	4	0.236	0.1632
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(1, "linear")	40	4	0.2769	0.2271

Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(128 units, "relu") Dropout(0.2)	20	4	0.0511	0.0879
Patrick Mahomes	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(256 units, "relu") Dropout(0.2)	20	4	0.0625	0.093

Running Back Model Statistics

Base Parameters						LSTM				
Player	**Season**	**End Training Week**	**Time Steps**	**Loss Function**	**Optimizer**	**Layers**	**Epochs**	**Batch Size**	**Loss**	**Validation Loss**
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2) LSTM (256 units) Dropout (0.2)	20	4	0.0952	0.2242
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2)	20	4	0.0797	0.1913
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	LSTM (128 units) Dropout (0.2) LSTM (128 units) Dropout (0.2)	20	4	0.0768	0.2047
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2)	20	4	0.0975	0.1509
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	LSTM (128 units) Dropout (0.2)	20	4	0.0647	0.1569

Base Parameters						RNN				
Player	**Season**	**End Training Week**	**Time Steps**	**Loss Function**	**Optimizer**	**Layers**	**Epochs**	**Batch Size**	**Loss**	**Validation Loss**
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(128 units, "relu") Dropout(0.2)	20	4	0.0254	0.1296
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(256 units, "relu") Dropout(0.2)	20	4	0.0082	0.2104
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2)	20	4	0.07	0.1577
Christian McCaffrey	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(128 units, "relu") Dropout(0.2)	8	4	0.0669	0.1527

Wide Receiver and Tight End Model Statistics

Base Parameters						LSTM				
Player	**Season**	**End Training Week**	**Time Steps**	**Loss Function**	**Optimizer**	**Layers**	**Epochs**	**Batch Size**	**Loss**	**Validation Loss**
Stefon Diggs	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2) LSTM (256 units) Dropout (0.2)	40	4	0.0632	0.0579
Stefon Diggs	2023	12	4	Mean Squared Error	Adam	LSTM (64 units) Dropout (0.2) LSTM (128 units) Dropout (0.2) LSTM (256 units) Dropout (0.2)	20	4	0.0651	0.0547

Base Parameters						RNN				
Player	**Season**	**End Training Week**	**Time Steps**	**Loss Function**	**Optimizer**	**Layers**	**Epochs**	**Batch Size**	**Loss**	**Validation Loss**
Stefon Diggs	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(128 units, "relu") Dropout(0.2)	8	4	0.0987	0.0497
Stefon Diggs	2023	12	4	Mean Squared Error	Adam	SimpleRNN(128 units, "relu") Dense(128 units, "relu") Dropout(0.2) Dense(128 units, "relu") Dropout(0.2)	10	4	0.0835	0.0659