1.0 Abstract

Fantasy football is a popular game where participants manage teams of real NFL players and compete based on their on-field performances. Accurate Fantasy Points Scored (FPS) prediction is critical for optimizing weekly lineups. This project leverages machine learning (ML) to enhance FPS projections, focusing on quarterbacks. Historical data was collected from ESPN and Yahoo APIs, preprocessed, and analyzed to identify key performance features, such as passing yards and touchdowns. A neural network model was developed using TensorFlow and Keras, incorporating three hidden layers and the RELU activation function.

The model achieved a Mean Absolute Error (MAE) of 4.41 and an R^2 score of 0.59, demonstrating its potential to support better player selection. While the results are promising, concerns about overfitting necessitate further testing with larger datasets. Future work aims to refine the model, explore hyperparameter tuning, and deploy the system as an automated lineup optimization tool. This study highlights the potential of ML in improving decision-making for fantasy football participants.

2.0 Background

Fantasy football is a game where participants create and manage a team using real NFL players, competing weekly against others in a league. Players score Fantasy Points (FPS) based on real-life performance, with the highest scorer winning the matchup. Participants set their starting lineup weekly by evaluating projected points and past performances, but projections don't always predict actual outcomes. Factors like location, weather, or team advantage can affect a player's performance, yet there's no easy way to account for them quantitatively. This project develops a Machine Learning (ML) model to improve FPS projections and support better player selection.

3.0 Data Collection & Analysis

Two data sets were collected and analyzed for this project. The first data was collected from ESPN for historical player and game data. While there are a lot of pre-collected data sets on the internet, many are behind paywalls. I wrote a program to query the ESPN APIs and collect historical player and game data. The second data set contains fantasy football points for this current season. I used a similar method to collect data from Yahoo's API. I selected Yahoo since I use the platform to host my league.

Once the data was collected, data preprocessing was performed. During this phase, I loaded the multiple CSV files collected from ESPN and Yahoo into pandas data frames. Since this project focused just on quarterback performance, I removed any data irrelevant to the quarterback position. I was also focused on ensuring data in the columns were the same type of data (e.g., float, int, etc.) and replacing any missing values with 0 where applicable.

Once the data was preprocessed, I turned to the analysis portion. I also ensured that if data for a player was stored across multiple data frames, I could reference a player ID to do an inner join during the project's feature engineering data augmentation phase. I leveraged heatmaps to help identify which features correlate most with fantasy football points scored. This confirmed my suspicions that passing yards and touchdowns had the highest impact on the quarterback's score. This should not be surprising since the fantasy football scoring system incentivizes quarterbacks to score through passing

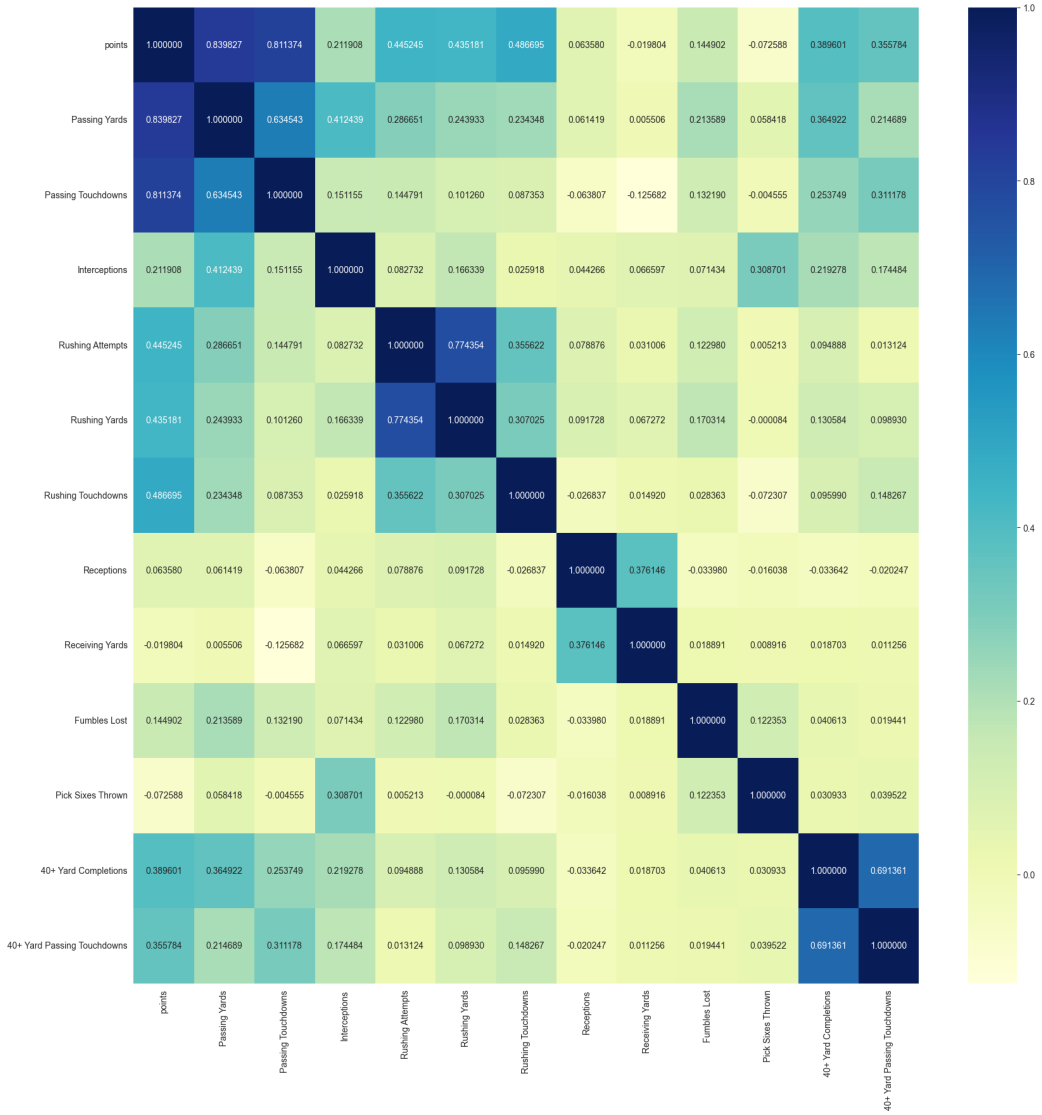

Figure 1: Player stats heatmap

4.0 Data Augmentation

Based on the identified correlations, I planned to build a data frame containing the various features I will use to train the model along with the fantasy football points scored in the game. I created a single data frame containing rows for each game during the last

season. Each row was a quarterback, their average stat values for each feature, and the Yahoo fantasy points they scored that game.

5.0 Model Selection

I selected to build a neural network. This was based on previous research[1][2] and what I had learned in class. I built a model with 78 features or nodes for the input layer with one node as the output layer. This output will represent the player's fantasy football points scored in the game. Method and Validation for Optimal Lineup Creation for Daily Fantasy Football Using Machine Learning and Linear Programming[2] mentioned adding multiple hidden layers to improve the accuracy of training models. This leads me to add three hidden layers of 78, 32, and 16 nodes each. These values were selected a priori and not backed by any specific reason. I did try various models by adjusting the number of nodes. However, I saw no significant changes to my model's performance metrics: mean absolute error (MAE) or R^2.

Now that we have defined the high-level model architecture, I needed to select an activation function. Based on research, I selected the RELU activation function. I also tried to use the swish activation function. I saw modest improvements in MAE and R^2 values. I used the standard scaler from sklearn with the Adam optimizer. Finally, I used the Mean Square Error (MSE) to calculate the error loss function. I used TensorFlow and the Keras API to build my model quickly. I selected this library based on the open-source community support and examples that I found online.

6.0 Training Methodology

Before training, I split my data set into a 90/10 split of training and testing data, respectively. I initially decided to do 1000 training epochs with a batch size of 8. I plotted the loss function to understand how the model is performing. One thing I noticed is the 1000 training epochs were not required. The graph below shows that the loss functions stabilize early.

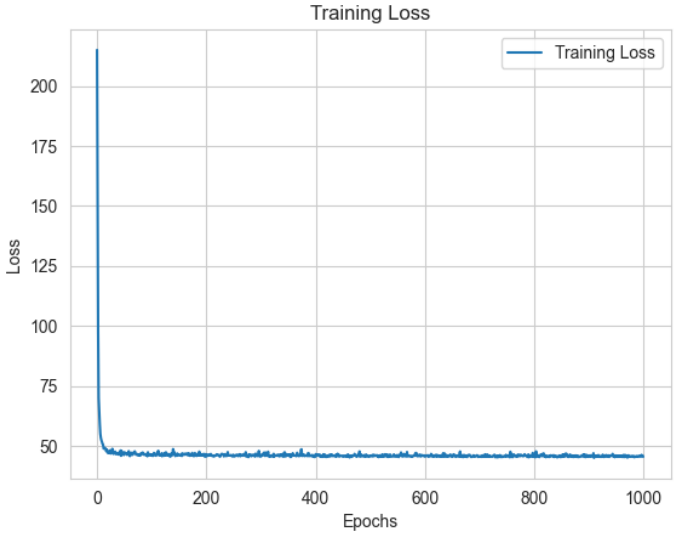

Figure 2: 1000 epoch training loss

After reviewing this graph, I adjusted the training epochs to 100. This confirms that I can further reduce the number of training epochs and still achieve similar MAE and R^2 values.

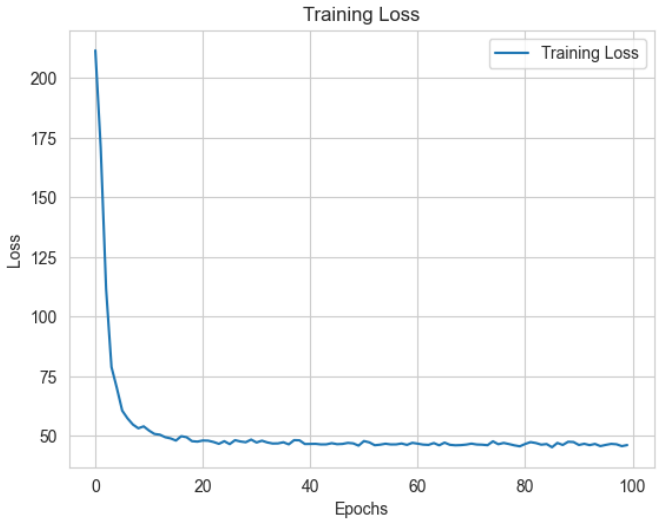

Figure 3: 100 epoch training loss

7.0 Results

Once I completed the training, I took the model and used the test data to validate the model. I used the TensorFlow predict function of my newly created model object feeding in the 10% of test data. I also collected the following metrics to understand the model's performance:

Mean Absolute Error (MAE): 4.414918416659036
Mean Squared Error (MSE): 37.0502057218576
Root Mean Squared Error (RMSE): 6.086888016208086
R-squared (R^2): 0.5947180986404419

I use the Mean Absolute Error (MAE) for this project to help me understand the average point prediction error. The next metric I used was the R2, which showed how well my model fit the training data. The fit score was great, and I would be happy with a +/- 4-point prediction score.

8.0 Future Work

The first thing I want to confirm or deny is whether my model is truly overfitting. Since this model was trained on a small data set, I am concerned it will not perform well with more extensive or future data sets. Based on this finding, I would adjust the model architecture or work on tuning hyperparameters.

Once I have confirmed the model is genuinely performing to our level of expectations, I would like to add this to an application for inference. Ideally, I would like to write a bot that can autonomously run my fantasy football team for me, including using a trained model to make player point predictions and set my lineup.

9.0 Stakeholder Acknowledgements

- Who are your stakeholders?
- How could they benefit from your model?
- How could your model have a negative effect?
- Are you able to have your stakeholders try your model? What do they think?

10. Conclusion

This project successfully demonstrated the potential of machine learning to improve Fantasy Football Point Score (FPS) predictions for quarterbacks. By leveraging historical data from ESPN and Yahoo APIs, preprocessing it, and engineering relevant features, we created a neural network model capable of predicting FPS with a Mean Absolute Error of approximately 4.41 points and an R^2 score of 0.59. These results indicate a reasonable level of accuracy, with predictions providing meaningful insights for lineup decisions.

The model architecture, including three hidden layers and the use of the RELU activation function, proved effective for this task, although hyperparameter tuning could further optimize performance. The findings also highlighted the need for additional data to address potential overfitting and validate the model's robustness across broader datasets.

Future work will focus on refining the model, incorporating larger datasets, and deploying it in a practical application to automate lineup decisions. This project illustrates the promising role of machine learning in enhancing decision-making within the Fantasy Football domain and lays a solid foundation for further advancements in this area.

Appendix A: Citations

[1] - Lutz, R. (2015). Fantasy Football Prediction. *ArXiv*. /abs/1505.06918
[2] - Mahoney, J. M., & Paniak, T. B. (2023). Method and Validation for Optimal Lineup Creation for Daily Fantasy Football Using Machine Learning and Linear Programming. *arXiv preprint arXiv:2309.15253*.