THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF APPLIED DATA SCIENCES

Fantasy Football Profitability: Using Machine Learning to Construct Optimal Teams and See Consistent Returns

ANDREW BAAK
SPRING 2021

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Applied Data Sciences
with honors in Data Sciences

Reviewed and approved* by the following:

Francis Wham
Data Scientist
Thesis Supervisor

John Yen
Professor of Data Sciences
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

Rapid technological development in computing over the past three decades has led to a rise in applying machine learning applications and methods to solve a plethora of problems across a variety of industries. The creation and expansion of the internet has also led to a completely new way to instantly access, share, and download information. Additionally, the sports betting industry has revolutionized as a result of the internet, which has led to the development of popular online platforms to place bets on numerous different types of sporting events, including fantasy football. These three technological developments have led to a rise in the study of developing machine learning models from the vast quantities of data available online to try and outperform the field of contest entries. Machine learning models can be developed to predict the optimal teams to construct for weekly fantasy football contests to consistently beat the fields and see monetary returns, much in the same way that mathematical models are used in stock prediction in the financial industry. A well-constructed model can enable a user to consistently profit from playing daily fantasy football. While the model created for this project did not end up producing profitable results, it serves as a solid foundation for this type of work going forward, and with improvements can likely produce consistent profits in years to come.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I'm extremely thankful to Dr. Wham for all his help and support as my thesis supervisor throughout this project. The insight and aid that he provided during our weekly meetings was invaluable and I would not have been able to complete this project without his help. I'm also thankful for his willingness to explore a topic for which I have a lot of interest and passion. I'd also like to thank my honors advisor, Dr. Yen, for his support and encouragement throughout this process. Additionally, I am thankful to my academic advisor, Susan Agee, for encouraging me to pursue a thesis. To my roommate, Phil White, thank you for your support and for providing a screenshot from your fantasy football league (of which he was a champion, I might add) to include in my thesis. To my friends in the Bermuda Triangle, thank you for your inspiration. Finally, I'm thankful to my mother for encouraging me to stick with this project and to complete my thesis.

Chapter 1 Introduction and Fantasy Football Background

With vast improvements in computation and the development of the internet, fantasy sports have transformed from their humble roots into a major industry, with millions of players around the world competing in various leagues focusing on many different professional sports (football, basketball, hockey, baseball, golf, etc.). According to a survey from the Fantasy Sports and Gaming Association (FSGA), there were an estimated 45.9 million fantasy sports players in 2019 in the U.S. alone, with the majority of them playing fantasy football [1]. As of 2015, fantasy football is an over $18.6 billion market, worth more than six times the average NFL franchise [2]. As fantasy sports have turned into a legitimate business, a variety of platforms have emerged for players, as well as several different types of contests in which to compete.

The focus of this work is on the playing of "daily" fantasy football, a particular format of fantasy football in which a new team is drafted each week rather than for an entire season. This format has become increasingly popular, particularly on mobile application-based betting sites in recent years. The goal of the present work is to develop a machine learning model which will aid in selecting the best potential daily fantasy football players at each position. This information can then be used to construct optimal daily fantasy football teams, which maximize the expected equity of entries placed into daily fantasy competitions.

First, some background and history on the subject of fantasy football and daily fantasy football will be provided. After that, related work in the field of machine learning for sports prediction will be examined. Next, an overview of machine learning and the methods intended for use in this project will be discussed. Additionally, the problem this project aims to solve will be formally defined, and the methods used to generate the data for training models will be

explained. After these methods are discussed, the models created will be reviewed and the results analyzed. Finally, this paper will conclude with a review of the work accomplished and outline the potential for future work related to the subject.

1. History of Fantasy Football

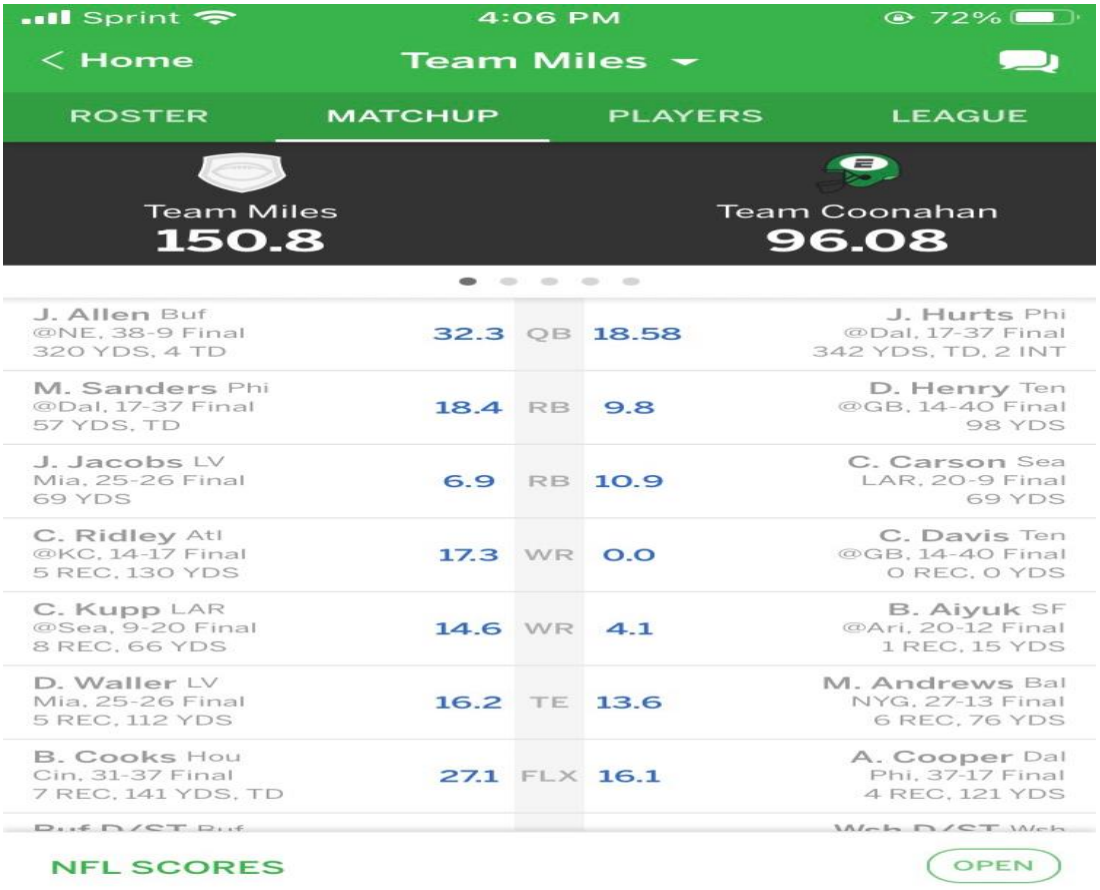
Fantasy Football has been around in some form since the early 1960s. California businessman Wilfred Winkenbach is credited with developing the first fantasy football league in 1962. With two other friends he devised the setup and rules for the Greater Oakland Professional Pigskin Prognosticators League. The format and rules of the league were spread through word of mouth and local leagues were formed. This early form of fantasy football never gained much traction, as the technological limitations of the 1960s and 1970s made it highly difficult to keep track of the necessary statistics to run a league [3]. Major advancements in computing, coupled with the boom of the internet, have finally created a consistent and reliable way to play fantasy sports. More sports-related data is generated than ever before and is also much easier to access and share. Platforms have been developed specifically for the purpose of playing fantasy sports.

2. Traditional Fantasy Football

Traditional fantasy football is played with one team over the course of a 16-game NFL regular season. NFL teams play 16 games over the course of 17 weeks, with one “bye week” for each team during the regular season for rest. Players are part of a league consisting of a set number of teams (most leagues have around 10 teams) and compete head-to-head each week. Rosters are drafted before the regular season of the NFL begins and kept throughout the course

of the season, though trades are allowed. Each week, a league member is matched against another league member and they construct a starting lineup out of the NFL players on their rosters. Most leagues require a team to include: One quarterback, two running backs, two wide receivers, a tight end, a "flex" position player (an offensive skill player like a running back, wide receiver, or tight end), a kicker, and a defensive unit. Fantasy players receive points based on how the players on their team perform in the actual NFL games played every weekend, and the fantasy team which accrues the most points in a given week wins.

Figure 1. Example of ESPN Fantasy Football Matchup [4].



3. Daily Fantasy Football

In daily fantasy football contests players draft a new team each week to compete in contests. Fantasy players also have a budget to work with each week and must put together a team that falls within budget constraints (called a salary cap). Different NFL players have different value, as dictated by their performance in previous weeks and throughout the season. As a general rule, high achieving players will cost more and low achieving players less. The goal is to maximize the potential points a team can score against the salary cap. Also, instead of playing in a single fantasy league throughout the course of a NFL season, daily fantasy players can compete in different types of contests each week as they are not tied to a league like in traditional fantasy football. Head-to-head competitions involve playing against a randomly selected opponent also enrolled in the competition. Whichever team generates the most points that particular week wins. Double Up contests are an extension of head-to-head play, where a larger field of teams is produced and the top 50% of players win double their entry fee. The final, and most profitable, daily fantasy contest is the weekly tournament. Participants can enter any number of teams for a fee and the top 20% of all teams entered in the tournament win some sort of prize, with the top prize often being extremely lucrative (upwards of $1,000,000) [5]. Fantasy sports are considered a game of skill rather than a game of chance, making them legal to play across the U.S, though there are a few states with exceptions.

The two main platforms to play daily fantasy are FanDuel and DraftKings, both relatively new sports betting companies. Prizes, salary cap, types of contest, and roster makeup are all slightly different on each platform. The below figures show what type of players each starting lineup must include, the salary cap for a team, and the targeted number of points needed to win on each platform.

Table 1. DraftKings and FanDuel platform comparison.

Platform	FanDuel	DraftKings
Salary Cap	$60,000	$50,000
Roster	▪ 1 Quarterback ▪ 2 Running Backs ▪ 3 Wide Receivers ▪ 1 Tight End ▪ 1 Kicker ▪ 1 Defensive Unit	▪ 1 Quarterback ▪ 2 Running Backs ▪ 3 Wide Receivers ▪ 1 Tight End ▪ 1 Flex ▪ 1 Defensive Unit
Targeted Points	120	150

Chapter 2 Related Works

Several previously completed papers examining the application of machine learning in sports and sports betting were consulted as reference for this project. They encompassed a variety of different problems related to machine learning and sports and used diverse approaches in their experimentation. This paper provided good background into the types of problems that machine learning can attempt to solve as well as frameworks that were successful in application.

1. ML Approaches to Competing in Fantasy Leagues for the NFL

In this paper, Landers and Duperrouzel experimented with different machine learning methods to attempt to predict the fantasy points that each NFL player would generate weekly, which could then be used to construct optimal teams that provide maximum points while meeting salary constraints. Their problem is similar to the one this project is attempting to solve. The experiment was broken into two separate but related problems: The creation of player fantasy prediction models through the use of machine learning and the optimization of these point predictions to create the best team possible. The methods of least squares and gradient boosted trees were used to create predictions for each player and data was drawn from the 2016 NFL season. The method of least squares was used as a baseline regression technique to compare to the later created boosted trees. After predictions were created, Landers and Duperrouzel worked on the optimization problem. Four different approaches with varying levels of sophistication for creating the best team possible were used and compared against each other. Upon completion of their experiment, Landers and Duperrouzel were able to create teams that produced point totals which were above the break-even threshold for competitions approximately 68% of the time [6].

Several ideas from Landers and Duperrouzel were discussed when planning the prediction and optimization problems for this project. Their use of boosted decision trees showed that using boosted trees to solve this type of problem was an effective approach to take and helped to support the choice of using XGBoost for this project. Additionally, one or more of the team picking strategies from Landers and Duperrouzel could be useful as a baseline to test against the team picking strategy used in this project. A final takeaway from Landers and Dupperouzel was their use of the break-even benchmark as a measure of success. Rather than purely striving to beat breaking even, for this project it was proposed that it may be more prudent to examine the expected utility of entries into a contest. Specifically, perhaps it would be better to examine the expected utility (what is this entry potentially worth vs. cost of entry fee) of each individual team entered into a contest and use expected utility to measure success.

2. The use of machine learning in sport outcome prediction: A review

This paper from Horvat and Job analyzed decades-worth of a variety of machine learning-related sports problems to survey the state of the industry at a high level. They examined a number of different previous experiments across several sports to uncover the common trends and discover useful takeaways that could be applied in future experiments. The paper produced several pieces of helpful information which were kept in mind and considered when conducting this project. A major takeaway was that neural networks tend to be the prevailing machine learning framework used on sports related prediction problems, regardless of the sport, though some sports utilized particular approaches more than others. An additional takeaway was oftentimes having more data did not correlate to better results. Horvat and Job

found that it was often better to have less data that accurately captured the current state of what event the experiment is attempting to predict. Experiments with vast datasets made up of years' worth of statistics were not outperforming experiments using smaller datasets by wide margins, though Horvat and Job acknowledge that it is hard to compare experiments across different sports which used different datasets even if the problems to solve were focused on the same sport. The paper also offered useful advice on methods to employ for feature selection such as principal component analysis, signal-to-noise ratio, and adding or dropping features as accuracy changed [7]. These methods were kept in mind when performing feature selection for this project. A final takeaway was that of the football-related experiments examined, the use of trees and/or a random forest method produced the highest average median accuracy for results, further supporting XGBoost as a potentially useful modeling approach [7].

3. Exploiting Sports-betting Market using Machine Learning

The authors of this paper experimented with convolutional neural networks (CNNs) to test the hypothesis that they could use machine learning methods to outperform traditional sports gamblers by "beating Vegas" consistently. They focused their study on National Basketball Association (NBA) betting from 2007-2014 and strictly looked at money line betting, where the bettor purely tries to determine which team will win a particular game. Logistic regression was used as a baseline for modeling and then a CNN was developed making use of NBA team and player data. The results of the experiment were promising, as their CNN outperformed the baseline logistic regression model, performed better than the average gambler purely placing bets off of chance, and in some cases the CNN was able to "beat Vegas". Through this research, the

authors of the paper also discovered that the betting strategy employed (how many bets to place, how much money to place on each bet, etc.) had a strong influence on the potential profitability of their model. A solid betting strategy proved to be equally important to a robust machine learning model. A final major takeaway from this paper was the discovery that profit on bets placed would decay when the model incorporated information from bookmakers/sportsbooks where bets could be placed. This decay is due to the fact that bookmakers/sportsbooks never offer truly even odds on a bet without a favorite. If the odds of either team winning a game are 50%/50%, the book will lower the odds on each bet (perhaps to around 48.5%/48.5%), introducing a margin which ensures they will profit regardless of the outcome [8].

Examining the strategies regarding betting explored in Exploiting Sports-betting Market using Machine Learning was helpful in thinking of devising betting strategies for this paper as well as framing the problem this paper attempts to solve in terms of gambling. Exploiting Sports-betting Market using Machine Learning was particularly helpful in not thinking of the problem that this paper attempts to address as purely a prediction problem, but rather prediction mixed with team optimization and solid betting strategy.

Chapter 3 Machine Learning and Methodology

1. Machine Learning Overview

Machine learning is a multidisciplinary field of study which combines elements of computer science, mathematics, and statistics. The field of machine learning has been active for decades but has only relatively recently started to see a variety of practical applications due to advances in computation. At its core, machine learning deals with trying to predict future outcomes based on past results and data. Training data (examples of previous results) are used to fit a model and develop a function that approximates an outcome as closely as possible. Machine learning can be broadly divided into three categories: Supervised learning, unsupervised learning, and reinforcement learning. This project focuses on the use of supervised learning. In supervised learning, training data includes a source target variable which the model then attempts to predict in the test set of data. For this project, the target variable to predict is the expected fantasy points a player will achieve in a week, and the training data is made up of statistical data for players and teams over several weeks' worth of games. This type of supervised learning is called regression, where the goal is to attempt to model the relationship between a dependent variable (predicted fantasy points) and number of independent variables (the various statistics generated by a player and team in a game).

A key component of machine learning is attempting to create a model which makes very accurate predictions, which is referred to as optimization. To "learn", the model attempts to minimize how wrong it is about the predictions that it makes. A loss function is used to calculate the inaccuracy of predictions made, and the goal of the model is to minimize the loss function, to make as robust predictions as possible. Mathematical methods such as differential calculus and

numerical analysis are used to find the parameters of the model that minimize loss. Different loss functions are used depending on the type of machine learning model being created. For this project, root mean squared error (RMSE) was utilized as the loss function. RMSE represents the square root of the summation across all training examples of the predicted value of a sample minus the actual value of the sample squared, divided by the total number of data samples in the training set.

Figure 2. Root Mean Squared Error Equation

$$RMSE_{fo} = [\sum_{i=1}^{N} (z_{fi} - z_{oi})^2 / N]^{1/2}$$

Zf_i = predicted value
Zo_i = observed value
N = number of observations

2. XGBoost Overview

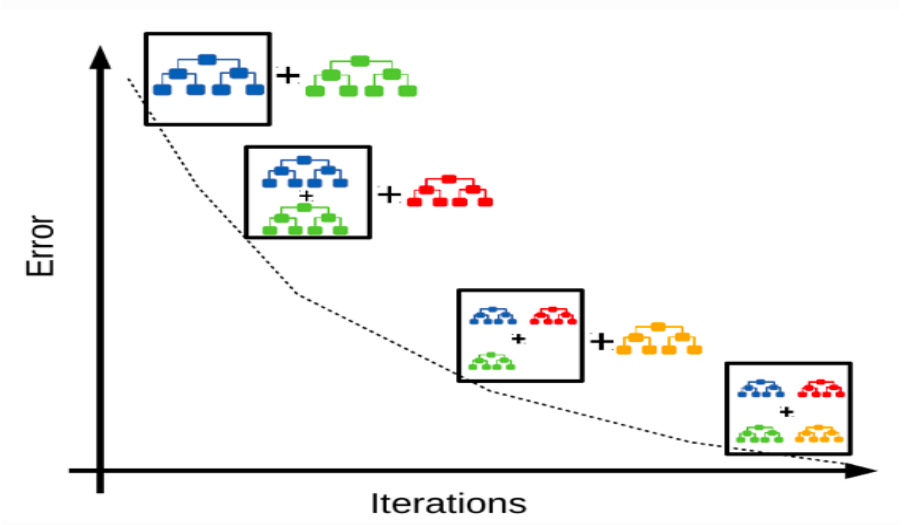
XGBoost (short for eXtreme Gradient Boost) is a machine learning library which has seen a rapid rise in use in the last half of the past decade. XGBoost is one of the predominant software libraries used for machine learning tasks involved structured or tabular data on classification and regression predictive modeling problems, under which this project falls. The approach is commonly used on the popular data science competition platform Kaggle with success. In 2015, 17 out of 29 winning solutions for competitions on Kaggle made use of XGBoost [9]. Additionally, the author of this paper has previously found success using XGBoost on sports-related machine learning problems, one focused on predicting the outcomes of NCAA March Madness Basketball games, and another project focused on the prediction of shot outcomes using

historical data from former NBA player Kobe Bryant. While other types of machine learning models and frameworks exist and have been used to solve sports-related prediction problems, XGBoost also brings the benefit of being highly interpretable and transparent as opposed to some other methods such as neural networks, which feature one or more hidden layers.

3. Tree Boosting Algorithms

XGBoost is based on the gradient boosted trees algorithm, with an emphasis on computational speed and model performance. Gradient boosted trees are a supervised machine learning method based on function approximation by optimizing a specific loss function and applying several regularization techniques. Several trees are constructed which are "weak" classifiers, which are then made better through the optimization of a specific loss function.

Figure 3. Visualization of a Tree Ensemble Algorithm [10]



Decision trees are another type of supervised machine learning algorithm where data is continually split according to certain parameters based on the information gain that each variable

in the dataset provides. Information gain is calculated for all variables and whichever variable has the highest information gain is used as the parameter to split data on. The process is then continued with the other variables until a full tree is created. Decision trees are made up of root nodes (where the data is split) and leaves (decisions/outcomes). Unlike decision trees, tree boosting algorithms make use of regression trees, which contain a continuous value on each leaf, as opposed to a class in which to be grouped [9].

To determine the best threshold to split a particular variable on, gradient boosted tree algorithms make use of greedy splitting algorithms which enumerate through all possible split thresholds on all the features in the dataset [9]. Trees are then constructed at a variety of thresholds, and hyperparameters are used to prevent overfitting of the model and to create the best model possible. Hyperparameters such as column sampling and row sampling are used to dictate what percentage of columns and rows respectively can be used in model construction, helping to prevent the model from overfitting. The model iteratively tries to correct errors made in previous trees, so that each new tree added improves upon the previous [11]. At the completion of the algorithm, an ensemble of trees is created, each new tree improving upon the performance of previous trees by finding parameters and splits which best minimize the particular loss function being used. Leaves in each tree correspond to continuous values which are predicted. To calculate the final prediction for a particular training example, the predictions from each tree are summed together [9].

4. XGBoost for Regression

First, an initial prediction is made for each row in the dataset, which is the average of the variable to be predicted (in this case fantasy football points). The residuals are then calculated for each of these data points. A residual is the actual value of a variable minus the initial prediction (in this case the average). After the residuals are calculated, a regression tree is fit to them. Next, the similarity score of all the residuals in a leaf is calculated by summing all the residuals, then squaring the sum, then dividing the sum by the number of residuals there are plus a regularization parameter, lambda.

Figure 4. Similarity Score Equation

$$\textbf{Similarity Score} = \frac{(\boldsymbol{\Sigma}\ \textbf{Residuals})^{\mathbf{2}}}{\textbf{Number of residuals} + \boldsymbol{\lambda}}$$

Following that, observations are split up by making more leaves and calculating the similarity scores of each leaf (using the same method described above). After splitting the observations and calculating new similarity scores, the effectiveness of the split is measured by calculating the resulting gain. The gain of splitting a tree into leaves is calculated by taking the similarity score of the leaf on the left plus the similarity score of the leaf on the right minus the similarity score of the root from which both leaves stem. If the gain is above a certain threshold, then the tree splits on that parameter.

Figure 5. Gain Equation

$$Gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

After that split, the threshold for the variable being used to split on is shifted to a different level, the tree is remade, and new similarity scores are calculated, repeating the previous process. After all the gains are calculated again at different thresholds, whichever split results in the highest gain is used in construction of the tree, becoming a branch. This process of determining the best way to split the tree into branches is then repeated for different variables in the dataset at different threshold levels until a complete tree is constructed.

A tree constructed can be pruned as needed by subtracting leaf gain from a user chosen tree-complexity parameter, gamma. If the calculated gain of a leaf minus gamma falls below zero, the branch which split the leaf in question is removed, otherwise it remains in the tree. This process is completed for all leaves in the tree. After pruning is completed, the output value for the tree is calculated by taking the residuals in a particular leaf and dividing their sum by the number of residuals plus lambda. Predictions are then calculated by multiplying the output of the tree by the learning rate and adding the initial prediction. New residuals are then calculated, and new trees can be constructed. The process is repeated as needed until the residual value becomes as small as desired or the maximum number of trees (a parameter which can be set) are generated. At the end, a series of trees are created which build on each other to create predictions. Each subsequent tree added improves on the previous trees, allowing for more accurate predictions [12]. Once this process is complete, the final tree constructed is used to generate predictions for the test dataset.

Chapter 4 Problem Definition

Making use of NFL data related to team performance, individual player performance, and player injuries, a model will be constructed to predict the expected fantasy football points that a player will produce in a week of the NFL regular season. Next, using the previously generated predictions, fantasy football teams will be constructed for each week of the 2017 NFL regular season which honor the constraints of DraftKings contests. Each team will be made up of nine players (one quarterback, one tight end, two running backs, three wide receivers, one flex position, one defensive unit), and the salary cap of the team will not exceed $50,000. To determine profitability, the actual points achieved by players on each fake fantasy team constructed for the 2017 season will be compared to historical DraftKings contest data from the 2017 regular season. If a constructed team in a given week surpasses the threshold required to win money in a particular DraftKings contest, the money won minus the cost of entering the contest will be added to the total profit. If the team in question fails to surpass the money-winning threshold of a particular contest, the cost of entering the contest will be subtracted from the total profit. The results of modeling and team construction will be deemed profitable if the total profit generated by constructed teams is greater than zero dollars, break-even if the total profit generated is equal to zero dollars, and unprofitable if the total profit generated is less than zero dollars.

Chapter 5 Data Integration

1. Data Sources

Four data sources were utilized for this project. The first major source of data was from the fantasydatapros Github repository which contained statistical data for NFL players. The repository included weekly player statistics from 1999-2019. The offensive statistics for any NFL player who played in a game during this twenty-year period were included. Files were broken down by year and week. For each year there were 17 individual comma separated value (csv) files corresponding to the 17 weeks of the NFL regular season. Each row in one of the csv files represented the statistics achieved by one player in a single game. Each row contained the player's name, position, the team they play for, as well as statistical attributes such as passing yards, rushing yards, rushing touchdowns, etc. [13].

The second data source came from Github user ryurko's repository and contained the box scores for every regular season NFL game from 2009-2019. 11 files existed, each one corresponding to one season. A row of data in one of the csv files represented a single game during the regular season of an NFL season. Columns in each row included a game identification number, the home team, the away team, the week and season in which the game occurred, the home team score, and the away team score [14].

The third data source utilized for this project came from footballdb.com and contains NFL player injury data. No csv files or other downloadable file formats were available, so a script was developed to scrape this data from footballdb.com and clean it up so that it could be merged with the other data sources. The dataset only contained injury data from the 2016, 2017, and 2020 NFL seasons as this information was all the data which was available on

footballdb.com. The data from all three years was combined into a single csv file after it was scraped. A row in this dataset represented the injury status of a single NFL player during one week of one season. Each row included the player's name, the type of injury they have, the type of practice they engaged in on Wednesday of the week in question, the type of practice they engaged in on Thursday of the week in question, the type of practice they engaged in on Friday of the week in question, as well as their status going into their game that particular week [15].

The final data source used for this project was taken from rotoguru1.com and contained historical fantasy football data from the popular daily fantasy platform DraftKings (discussed in Chapter 1, Section 3). The website contained semicolon separate value (scsv) format of the data on each page, but still had to be scraped as there was no option to export it. The website included historical DraftKings data from the 2014 – 2020 (present day) seasons. A row in each table corresponded to the information and results of a single player's performance in one week of one season. Each row in a table included the player's name, the week and year of the game in which they played, the game identification number, the team the player was on, the opponent they played, whether they were on the home or away team in said game, the position of the player, the fantasy points they earned in said game according to DraftKings, and their daily fantasy football salary for said game as generated by DraftKings [16].

2. Integration Methods and Feature Engineering

After the raw data sources were compiled, data wrangling and feature engineering scripts were developed to get the data into model ready format. First, a script was developed to process all of the raw box score data. Each file containing box scores was iterated through, appended to a

single data table where all the columns of interest were extracted, and then written out to a single csv file called allBoxscores.csv.

A similar script was then created to process all the player related raw statistical data. Each week file within each year folder was iterated through, combining all player related statistics into a single data table which was then written out to a single csv, gamePlayerstats.csv.

The next script developed combined the box score data and player statistical data into a single file. Shared columns between the two tables were used to merge them together, and variance statistics were generated for such categories such as passing related data and rushing related data. The finalized file was then written out to a csv, StatsandBoxscores.csv.

After the development of StatsandBoxscore.csv, two scripts were developed to perform rolling merges on player related and team related statistics, to capture the element of time. A list of statistical values was generated, which was then iterated through and a rolling merge was performed on each statistic, summing each statistic over a certain number of weeks which could be specified by the user. After these two scripts were run, a teamallowed.csv was generated which included all the statistics that a particular team allowed up to a certain number of weeks as specified by the rolling merge, and a playerachieved.csv which included all the statistics that a particular player achieved up to a certain number of weeks as specified by the rolling merge.

Next, a script was written to scrape all the injury related data from footballdb.com. First, a list of all possible url paths was generated and iterated through, all merged into a single data table. This data table was then converted to a list, and each element in the list was read using the read_html function, and the relevant data extracted from each table on each webpage of interest and stored in a new data table. This data table, which contained all injury data, was then cleaned up. The formatting of certain columns was changed to make them easier to merge into other files

and easier to read. After formatting, dummy variables were manually generated for all the possible injuries that a player could have. These dummy variables were merged to the data table of injury data, which was written out to a csv, injuryreports.csv.

The next script merged together all necessary data into a training data file which was ready for modeling. Player achieved data and team allowed data were first merged together. Columns not of interest for modeling were dropped from each data table, the two tables were merged together on shared columns, and value difference columns were computed. This data was then merged with the previously generated injury data to create the final training dataset. Again, shared columns were used to merge the two data tables together, and the injury data was merged into the combined player and team data. All of this merged data was then output into a train csv, ready for modeling.

Though independent from the data required to run model training and testing, a script was developed to scrape the DraftKings historical data from rotoguru1.com. A list of all possible url paths was generated and iterated through. Next, a data table was created containing all the data scraped from the individual webpages corresponding to each url. This data table, final, was then cleaned up and formatted to make it easier to merge into other data for later use. The order of player name was changed from last name, first name to first name, last name so it would be the same as other columns in other tables. Team abbreviations were changed to the same format as abbreviations in the training data. Finally, a new column, points per dollar (PPD) was created, which took each player's DraftKings fantasy points and divided them by their salary that week, to see which players were most valuable at each position. Once the cleaning was completed, this data was written out to a csv, DKsalaries.csv.

Initially, all of the scripts were generated independently and run separately. To speed up and streamline the process of generating training data, the decision was made to merge all of these scripts into a single pipeline, called optimizedfulldatawrangling.R. This singular script could then easily be run from the command line, where the number of weeks to perform a rolling merge over could be set as a parameter. Running this script resulted in the development of five distinct training csv files being developed, using rolling merges of three weeks, four weeks, five weeks, six weeks, and seven weeks, respectively.

Chapter 6 Model Training

1. Model Overview

Upon the completion of data integration and feature engineering, the focus of the project shifted towards model building and parameter tuning. Several iterations of modeling scripts were created until a finalized testing script was settled on, fullmodeltesting.R. This script was automated so it could be easily run from the command line, with two parameters specified by the user. The first parameter, roll, determined which training data set to use corresponding to how many weeks the data was rolled over (three, four, five, six, or seven weeks). The second parameter, position, set which player position to train and predict on (quarterback, wide receiver, tight end, or running back). Different model hyperparameters were used throughout training to determine which ones would produce the best model. Throughout the process, the model was run for each position (four total), with each set of training data (five total), with all hyperparameter combinations (12 total) for a total of 240 iterations. The results produced will be discussed in later sections.

2. Data Preparation for Modeling

Before each XGBoost model could be created and have the data fit to it, the training data first needed to be manipulated into a format compatible with XGBoost. First, the training dataset was read in and filtered so that only the player position of interest would be examined. Train and test sets were then created by sampling the data by year. Data from the 2016 season was used to train the model on and data from the 2017 season was used for testing, as these were the only

two years in the dataset which contained complete information from all four data sources (player statistical data, box score data, player injury data, DraftKings contest data). Next, null values were removed from the dataset as well as columns with little to no variance, as they would not end up being useful in tree construction. To handle categorical data, dummy variables were created. Dummy variables encode categorical data (such as practice status) in a numerical way so that it can still be interpreted by a model. Finally, the training and testing datasets were both converted to matrices, as XGBoost requires input data to be in matrix format.

3. Hyperparameter Tuning

An important part of any machine learning project is discovering which model parameters produce best performance. As each machine learning project goal is different and makes use of different datasets, the best parameters for a particular model are unique. For this project, a csv file of 12 different hyperparameter combinations was used to determine which combination would work best for the XGBoost model. The hyperparameters to be tested included gamma (a tree complexity parameter that serves as a threshold of whether a tree should split on a certain parameter or not), eta (the learning rate, which reduces the risk of model overfitting), max_depth (the maximum depth a tree can built to), min_child_weight (minimum sum of instances in a leaf node needed for it to be kept in the tree), subsample (percentage of training data which can be used in tree construction, helps to prevent overfitting), and colsample_bytree (percentage of columns in the data which can be used in tree construction, helps to prevent overfitting) [16]. The 12 different hyperparameter combinations were iterated through using a for loop, and cross-validation and training was run on the training data for each

hyperparameter set. RMSE scores were kept track of throughout each run so the best hyperparameters could be determined based off of which combination produced the lowest RMSE values during cross validation and training.

RMSE scores were recorded for each iteration of the hyperparameter list, across all positions and sets of training data so that they could be compared later to determine which combination was best. All results from the model training were merged into a Microsoft Excel workbook where they could easily be compared to one another. The best hyperparameter combination was selected by picking the one which most often produced the lowest RMSE across different positions and training data sets.

Table 2. Best Hyperparameter Combination

Parameter	**Value**
Objective (objective function)	Reg:linear
Gamma	0.1
Booster	Gbtree
Eval_metric	Rmse
Eta (learning rate)	0.1
Max_depth	2
Min_child_weight	27
Subsample	0.9
Colsample_bytree	0.9
Tree_method	hist

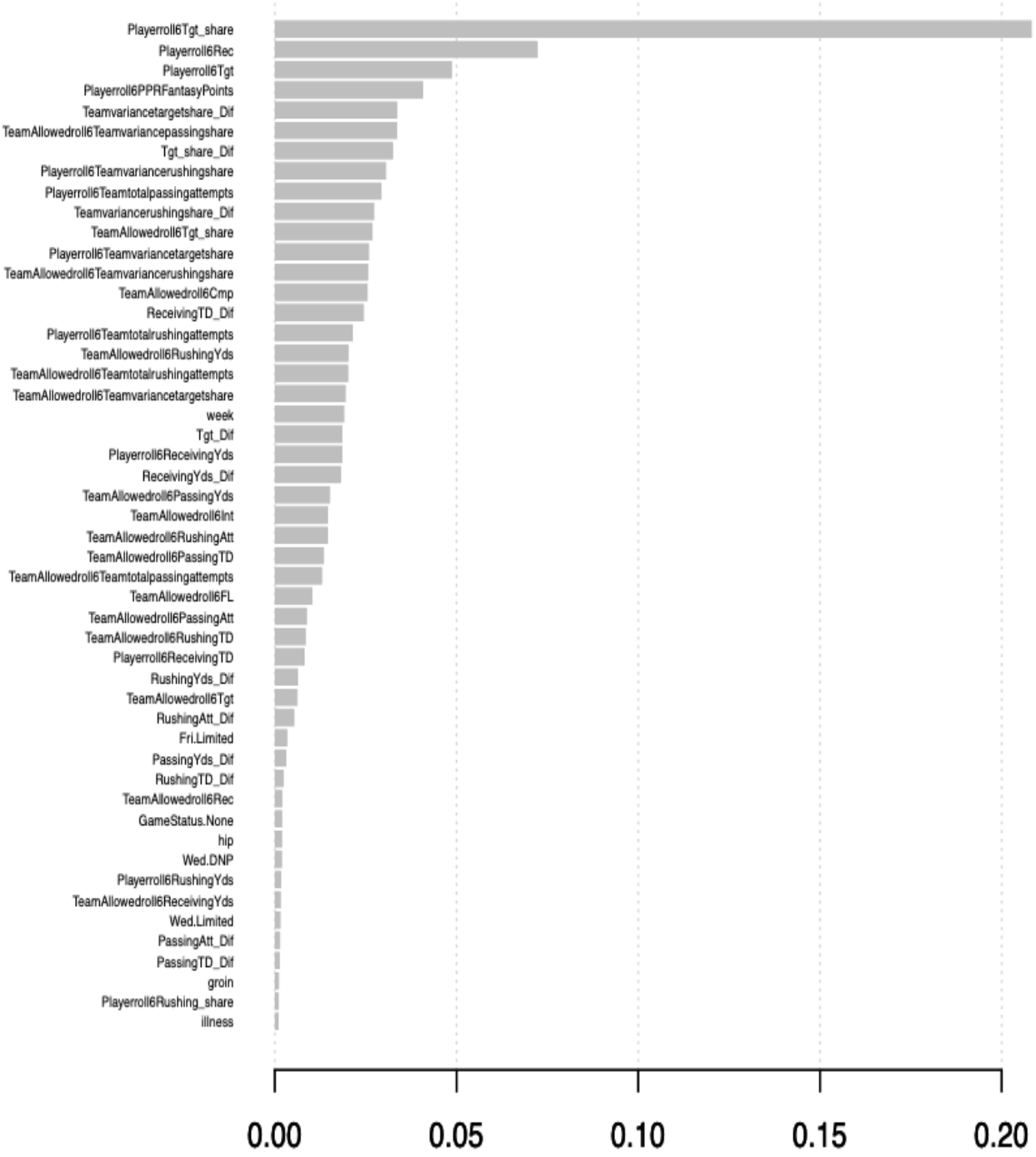
4. Feature Importance

Another crucial step taken during the model training process was examining feature importance plots for each ensemble constructed. Feature importance plots are a graph which can be generated from many tree-based machine learning algorithms which show the programmer which columns from the training dataset are most commonly being used in tree construction. These plots can be used to determine which columns are most integral to creating a tree. Feature importance plots were used in this project to aid in dimensionality reduction, simplifying the model and helping produce better results.

When injury data was added into the scope of this project and combined with the other data sources to create a training dataset, it greatly increased the dimensionality of the data. Over 80 distinct columns corresponding to injuries were added to a dataset which already had close to 100 columns. Having too many features in a dataset can make modeling very difficult, as it becomes harder to determine which features are actually playing an important role in tree construction, and also can often lead to overfitting [17].

To prevent this problem from occurring, feature importance plots were generated each time the model was run, and they were saved so they could be examined later. After all 240 runs of the model were completed, the plots for each player position and set of training data were generated and examined to determine which injury columns were appearing most often in the trees constructed and thus were important in modeling. Similar to what was done during hyperparameter tuning, all of the injury columns appearing in a particular model iteration were merged to an Excel workbook where they could be compared. The top nine injuries appearing most commonly across all player positions and rolling merges were kept in the dataset, and all others discarded.

Figure 6. Roll Six Wide Receiver Importance Plot



Chapter 7 Model Testing

At the conclusion of model training, the project moved into the testing phase to determine in the predictions made were accurate and could be used to enter DraftKings daily fantasy contests and produce profits. The first step was to take the individual predictions generated for each player and use those to construct teams which conformed to DraftKings requirements.

1. Team Construction

After predicted point values were generated for each player in the testing dataset, a script was developed to put teams composed of these players together to analyze how they would have performed in real DraftKings contests. Two functions were used to accomplish this task. The first, getPlayers, generated a list of all the possible combinations of players for each position in the dataset fed to it. The dataset consisted of the predicted fantasy point values in a given week of the 2017 season for a player, the actual fantasy points scored in that week by the player, their DraftKings salary in that week, their position, and their team. Different positions had a different number of maximum players that could be grouped together depending on how many players of each position were needed in a starting lineup as dictated by DraftKings. For example, getPlayers generated a list of quarterbacks with only one quarterback in each row as there is only one quarterback in the starting lineup of a DraftKings contest, but for wide receivers and running backs, the list often contained three or four players in each row, as these positions feature more than one player in the starting lineup. A salary cap variable which could be manually set by the

user was also used in getPlayers to ensure that each list of players in a position generated did not spend too much money, an important parameter to keep in mind when putting teams together.

The second function used, teamConstruct, used the output from getPlayers to generate a complete team by iterating through all the lists of players at different positions and constructing a team which maximized the potential points to be scored while making sure that the teams constructed stayed under DraftKings salary cap ($50,000/team). To start, getPlayers was run for each position group of interest (quarterbacks, wide receivers, tight ends, running backs) so that all possible player-position combinations could be compared. Teams were constructed by taking each list generated by getPlayers and selecting the group for each position which had a high potential to generate a lot of points (as determined by each player's predicted points value generated by the model) and kept the team salary cap below the $50,000 threshold. A for loop was then used to iterate through each week of the 2017 NFL regular season, and teamConstruct was run to generate a daily fantasy team for each week resulting in the creation of 17 teams.

2. Data to Test Against

To test whether or not the teams generated by the model and teamConstruct were financially successful, historical DraftKings results were manually pulled from rotogrinders.com, an online database which contained the results of historical fantasy football contests across several platforms. Testing for this project focused on DraftKings Double Up contests from the 2017 regular season. In a Double Up contest, players enter a team into the contest for a certain entry fee, and if they perform better than 50% of the field, they win double their money. DraftKings ran 624 Double Up contests during the 2017 regular season, which worked out to

approximately 37 contests per week. A row in the historical DraftKings dataset contained columns such as the name of each contest, the week of the season in occurred in, the prize pool, the contest buy-in cost, the top prize, the maximum number of entries a player could submit to the contest, the total number of entries, the points a team needed to generate to win (called the cash line), and the top score achieved in each contest.

3. Testing Results

To determine the success of the model, the DraftKings data was first merged to the data generated by the teamConstruct function, using week as the variable on which to merge. This function produced a table (called analysis_table) which had all the data stored in the DraftKings data table, as well as the predicted points and actual points which would have been generated by each team created by the teamConstruct function. To determine profitability, a new column, profit, was added to the analysis_table. Profit was calculated by comparing the actual fantasy points each team would have generated to the cash line point number from DraftKings. If the fantasy point total of a team was greater than the cash line of a particular contest, the top prize column in each row was subtracted from the buy-in column to calculate the amount of money which would have been made from entering the contest. If the fantasy point total of a team was less than the cash line of a particular contest, the negative value of the entry fee was added to the profit column for that particular row to indicate the money which would have been lost by entering the contest.

After this process was completed for each row in the analysis_table, the total profit resulting from testing was computed by summing the profit column across the entire data table. For the 2017 NFL regular season the profit generated was -$11,086.

Chapter 8 Discussions

While the model created for this project was not profitable, there is no reason to be discouraged or concerned moving forward. The limited timeline which this project was completed on, as well as not having quality data sources to make use of, likely accounted for poor performance in this initial iteration. This model still shows great potential going forward, with several avenues for improvement which could result in consistent profitability in the future. The foundational framework and underlying methods are solid but making changes to some of the details will likely result in better performance.

One of the main areas of improvement for the model is the data which was used in training and testing. Due to limited time and lack of high-quality open source datasets, all of the data used from this project was either scraped or pulled from a variety of asynchronous sources. This lack of synchronous data made data integration challenging and limited the amount of data gathered which could be used because some of the data gathered only had records for a small number of years. For instance, due to lack of overlap between the statistical player-related data and the injury data, the model was trained and tested on a very limited snapshot of historical data. With more time, the code written for this project could be easily converted over and make use of better data sources, which would likely help to improve model performance.

An additional area of the model which would benefit from more development and data is the data related to player injuries. Deriving a feature out of that dataset which accounts for when the teammate of a particular player is injured and unlikely to play in a game in the coming week would be useful. This feature would be principally helpful when two teammates play the same position. For example, if the best wide receiver on the Green Bay Packers is injured and will miss a game during week eight of the 2021 NFL season, it is likely that the Packers' second-best

wide receiver will have an increased number of passes thrown his way, as he becomes the temporary best wide receiver for his team that week. The increased passes thrown his way creates the potential for this player to generate more fantasy points than he normally may, resulting in a better performance. Additionally, due to the fact that this hypothetical player is typically viewed as the second-best receiver on his team, his fantasy salary for the week in question is likely to be significantly lower than other wide receivers in the league who are considered “stars”. A player such as this would likely be beneficial to add to a fantasy team, as they have high potential upside at a lower cost than many players in their position group, resulting in a high potential benefit per cost.

A final area of improvement to make to the model going forward is the incorporation of weather data. No weather data was incorporated into this initial model due to a lack of time but adding it going forward could also benefit model performance. Weather can play a large role in the strategy a team employs, and which players are likely to be more involved. If the weather in Massachusetts on the day the New England Patriots and the Cleveland Browns play is forecasted to be very windy and rainy, it is unlikely that either team will throw the football very much and will instead focus their offensive plays on running the ball. This weather would result in increased opportunities for running backs on both teams, while quarterbacks would not throw the ball as much and wide receivers would see less balls being thrown their way, all of which would have a major impact on the fantasy performance of each player involved in the game.

Chapter 9 Conclusion

This paper provides an overview of a project undertaken to model the relationship between an NFL player's performance, NFL team performance, injury data, and the fantasy football points they are predicted to generate in a particular week of an NFL season. First, data was gathered and scraped from a variety of sources across the internet. This data was then merged into a central code repository, where it could be cleaned up and integrated together. After this merger, model training was performed to determine the best parameters to use for the XGBoost regression model, as well as which particular injuries should be kept in the model for testing. Next, model predictions were made for data related to the 2017 NFL regular season, which were then used to construct teams for DraftKings Double Up Daily Fantasy Football Contests. 17 teams were generated, one for each week of the regular season. Model performance was measured by determining the profit that each team constructed would have been able to generate in the contests into which it could have been entered. The summation of profits across all contests run by DraftKings during 2017 for all teams generated by the model resulted in losses of $11,086. While the model was not profitable in this initial iteration, there is potential to make improvements and potentially see profitability in the future.

Though the final result produced was not as desired, this project remains an extremely beneficial undertaking for me. I learned how to run a data science project from start to finish, how difficult data integration across sources can be, the need to conduct robust model training, how to develop machine learning pipelines, and much more. While this project did require a lot of time and effort, it was a valuable undertaking, and I will keep the lessons I learned with me going forward.

REFERENCES

[1] FSGA Staff. “Industry Demographics”. Fantasy Sports & Gaming Association, https://thefsga.org/industry-demographics/.

[2] Sports Management Degree Hub Staff. “The Lucrative and Growing Fantasy Football Industry”. Sports Management Degree Hub. https://www.sportsmanagementdegreehub.com/fantasy-football-industry/

[3] J.L. Seto. “Who Invented Fantasy Football?” *Sportscasting -Pure Sports*, Sportscasting, www.sportscasting.com/who-invented-fantasy-football/.

[4] Figure 1. Example of ESPN Fantasy Football Matchup. White, Phil. 2020

[5] “Fantasy Football Quick Overview”. *How to Play One Week Fantasy Football*, DailyFantasySports101, https://www.dailyfantasysports101.com/football/.

[6] J. Landers and B. Duperrouzel. “Machine Learning Approaches to Competing in Fantasy Leagues for the NFL,” in *IEEE Transactions on Games*, Vol. 11, pp. 159-172, 2019.

[7] T. Horvat and J. Job. “The use of machine learning in sport outcome prediction: A Review,” in *WIREs Journal on Data Mining and Knowledge Discovery*, Vol. 10, Issue 5, 2020.

[8] O. Hubacek, G. Sourek, F. Zelezny. “Exploiting sports-betting market using machine learning,” in *International Journal of Forecasting*, Vol. 35, Issue 2, pp. 783-796, 2019.

[9] T. Chen and C Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, pp. 785-794, 2016.

[10] Figure 3. Visualization of a Gradient Boosted Trees Algorithm.Vasiloudis, Theodore. 26 August, 2019. http://tvas.me/articles/2019/08/26/Block-Distributed-Gradient-Boosted-Trees.html

[11] Dayananda. “How does XGBoost Work?”. *Towardsdatascience*. https://towardsdatascience.com/how-does-xgboost-work-748bc75c58aa

[12] J. Starmer. StatQuest with Josh Starmer. (2019, December 16). *XGBoost Part 1 (of 4): Regression* [Video]. Youtube. https://www.youtube.com/watch?v=OtD8wVaFm6E

[13] NFL player performance statistical data from 1999-2019. Github, fantasydatapros. https://github.com/fantasydatapros/data

[14] NFL regular season box score data from 2009-2019. Github, ryurko. https://github.com/ryurko/nflscrapR-data/tree/master/games_data/regular_season

[15] NFL weekly player injury data from 2016, 2017, and 2020 seasons. The Football Database. https://www.footballdb.com/transactions/injuries.html?yr=

[16] DraftKings historical daily fantasy football results data from 2014-2020. Rotoguru. http://rotoguru1.com/cgi-bin/fyday.pl?week="

[17] "XGBoost Parameters". *XGBoost Parameters – 1.4.0-SNAPSHOT Documentation*, XGBoost. https://xgboost.readthedocs.io/en/latest/parameter.htm

ACADEMIC VITA

Andrew Baak

Education

Pennsylvania State University — University Park, PA
College of Information Sciences and Technology | B.S. Applied Data Sciences — May 2021
College of Information Sciences and Technology | Minor in Security and Risk Analysis

Professional Experience and Research

ZTech
Software Engineering Intern — May 2020 – July 2020
- Worked with a small Austin based startup on an all-intern team to create a web application
- Designed database and wrote SQL commands for table creation
- Wrote code to aid in querying database from the backend
- Researched and created documentation for various APIs to use in the application

Deloitte Consulting LLP
Business Technology Consulting Intern — June 2019 – August 2019
- Worked on a data center migration project for a Fortune 500 financial institution
- Provided direct support to the external connectivity workstream
- Created internal documentation to aid in cost forecasting and onboarding
- Aided in hardware purchase tracking and validation

Johnson & Johnson
ITLDP Co-Op — June 2018 – Dec. 2018
- Worked on the Consumer Digital Marketing Team supporting the global website build process as a service specialist performing high-level project kickoff and oversight
- Created process help guide to onboard developers and testers to a new software platform
- Improved consistency initiatives in the website build process between clients and vendors
- Worked as an assistant Scrum Master and Android developer on the Health Partner Knees and Hips application

Nittany Data Labs
Research Member — Sept. 2017 – Dec. 2017
- Worked with student run data science club which completes research projects with various companies and universities
- Completed team project focusing on NFL twitter sentiment analysis using Python scripts and natural language processing
- Chosen as one of the top four projects from the semester
- Presented findings to the organization and faculty from the College of IST

Leadership Experience

Penn State Men's Club Volleyball Team
Secretary — May 2019 – May 2020
- One of four members of the executive board who communicate with the Club Sports Office
- Made decisions regarding which players to add to the team and what strategies to employ
- Supervised team communication and acted as an intermediary between club and parents
- Planned and supervised team community service events
- Oversaw the creation and distribution of team merchandise