Gridiron-Gurus Final Report

Kyle Tanemura, Ryan McKinney, Erica Dorn, Michael Li
Senior Project
Dr. Alex Dekhtyar

June, 2017

Contents

1 Introduction

With this project, we set out to create a web application that could allow fantasy football players to play with the assistance of, and against, artificial intelligence. Our application Gridirion Gurus allows players to create accounts, start a fantasy league, draft players, choose what players to start, and shows their fantasy team's performance throughout the season. The AI assists the player by recommending NFL players to draft, and by recommending which players to start on a given week. The fantasy player also has the option of customizing the AI bots general strategies, and even allow the AI to take the place of a fantasy player in their league.

We predicted NFL player's 2017 season performance using Python's commonly used data science modules such as Scikit-Learn and Numpy. Regression models were built using NFL player data from 2008 - 2016, and player's 2017 performance was predicted using their 2016 season's statistics. The backend of our application, including the implementation of the AI, was written in Javascript. We stored our data in a remote Firebase database and wrote our application using Vue and Electron as our frameworks.

2 Player Performance Prediction

2.1 Components of Prediction

For predicting fantasy performance of a player, there are two major factors to predict: how often a player will get the ball and how well he will do with it. Specifically in terms of fantasy sports, the number of points a player gets is highly correlated with the amount of times that player is involved in a play. A player's fantasy points usually do not take into account the number of times they get the ball, so a running back who runs the ball 3 times for 5 yards each will get the same amount of points as a running back who runs the ball once and gets 15 yards. This makes predicting the number of plays just as, if not more, important than predicting how many yards that player will get in that play. There are exceptions to this; an example being PPR leagues, where players earn points for every successful catch, regardless of whether the play is an overall positive or not.

We decided to have a similar approach to both problems, with a few key differences. We used multiple regression to predict both, with the explanatory variables being various statistic from only the player / team's previous season. This was because predicting players with varying numbers of previous seasons would not be very compatible with regression, as a regression model has to be fitted with data with no missing values. Another reason is the intuition that the most previous season is, by far, the best predictor of the next season's performance.

In order to make experimenting with and building models easy, we built a data pipeline in python that would create a regression model given the variable to predict and the desired explanatory variables. It extracts the explanatory variables from the previous season's statistics and builds a pandas dataframe. The pipeline then uses those to create a model on training data, tests it against the training data and displays the correlation coefficient between the predicted and actual. The pipeline supports interactions, polynomials, and categorical explanatory variables (and any combination of the three).

2.2 Predicting Playing Time

As stated previously, one of the major factors in the number of fantasy points a player will score is his play time. To do this, we predicted two variables: the number of games a player will play, and the number of plays per game he is involved in. The statistics we had at the time did well for predicting the numbers of plays a player will get in a game, but did poorly to predict that number of games they would play.

This was because any changes in the amount of games played from their previous season could not be detected. To more accurately predict play time, we scraped another dataset containing whether every player was a first string (starter), second string, etc for that season. We then used this as a categorical variable to hopefully better predict number of games played in the season. There was an improvement but it wasn't significant, so we instead chose to use a single categorical variable that was a combination of their previous roster rank and this season's. This way, the model could look at a player going from third string to first string different than a player staying at first string, which the previous method would not pick up. This greatly improved the accuracy for predicting number of game snaps played.

2.3 Predicting Player Statistics

```
# An example of values predicted for a single player
{
  "Player" : "Drew Brees",
  "Position" : "QB",
  "PredAvgPassYrds" : 8.0192111172,
  "PredAvgRushYrds" : 0.9649175544,
  "PredFantasyPoints" : 312.4552471505,
  "PredFumblesPerGame" : 0.2040063787,
  "PredGamesPlayedPercent" : 0.9346685774,
  "PredInterceptionRatio" : 0.0312153668,
  "PredPassAttPerGame" : 41.7119150136,
  "PredPassTDAttRatio" : 0.0527109235,
  "PredRushAttPerGame" : 1.8003801976,
  "PredRushTDAttRatio" : 0.023627895,
  "PredSeasonFumbles" : 3.0508536288,
  "PredSeasonInterceptions" : 19.4718042985,
  "PredSeasonPassTDs" : 32.8804974522,
  "PredSeasonPassYrds" : 5002.2961666257,
  "PredSeasonRushTDs" : 0.6361607711,
  "PredSeasonRushYrds" : 25.9795760632,
  "Season" : "2017",
  "Team" : "NO"
}
```

The number of fantasy points for each statistic of a player is entirely up to the players controlling the fantasy football league, and we wanted to make sure our application could work with a customizable scoring method for players. Because of this, it made much more sense to predict a player's statistics for the coming season, and then calculate their projected fantasy points from those predictions, rather than attempting to predict the number of points a player will earn using a

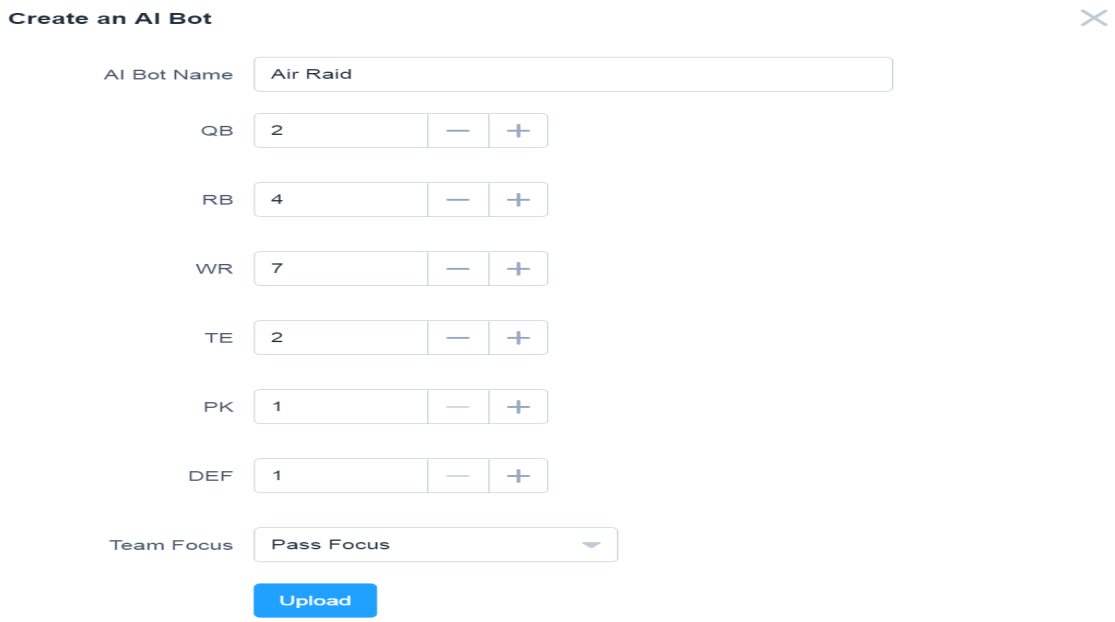
standard scoring rule set.

Every position had a separate set of models, with each of those models being a statistic to predict. Each model was created with the pipeline through experimentation. The options for explanatory variables were any of the player's previous season statistics, the roster ranking for each season, and any statistics associated with the offense of the team the player was playing on. Other options include polynomials and interactions between any of the variables just listed. Every variable was normalized to a per-game basis to allow the models to easily be updated a single game's worth of stats. This way the models could update a player's predicted performance during the season with his current season's game statistics. This was important because we wanted our application to identify rising stars as well as potential busts as a season progressed.

3 Artificial Intelligence

3.1 AI Profiles

The main function of Gridiron Gurus is the ability to create different "AI Profiles" that can be used by both the season long and week to week fantasy football prediction algorithms. Each profile contains a team composition it considers ideal, and a "team focus" which is a general strategy on how to gain points. Both affect the weight given to the various positions in slightly different ways. The ideal composition gives an AI profile goals to try to fulfill when constructing a team. The team focus affects more of the quality of each player that is taken. An AI profile with a team composition containing many RBs, few WRs, and few TEs, but, a passing focus for instance will most likely produce a team with many low level RBs and a few WRs and TEs that the algorithm considers of higher tier.



3.2 The Draft AI (Season Long Prediction)

A fantasy football draft takes place before the season starts, and involves participants taking turns on which players to include in their fantasy team. Standard rules are participant's starting roster must consist of 1 QB, 1 TE, 2 RBs, 2 WRs, 1 K, 1 DEF, and 1 flex position that can be either a RB or WR. A participant also has a bench of 6 players that are available to swap but don't earn any fantasy points. We set out to create an artificial intelligence that suggest 3 players to draft at every round. Our fantasy predictions allowed us to have projected fantasy points for every player who played in the 2016 season. The main challenge in creating a draft AI is in picking which position to draft at the current round. Once the AI knows this position, it is simply a matter of picking the one with the highest projected performance.

To do this we needed a metric that would indicate how much the AI wants to draft each position at this round. To assist in calculating this metric, the AI takes in two inputs from the fantasy player: his desired team distribution at the end of the draft, and a general playstyle indicating which positions he wants to draft more aggressively.

The AI then takes this information and looks at the distribution of projected fantasy points in the top players that are still available to be drafted for every position, and groups them into tiers. It looks at the number of players in each tier of each position and calculates a modified version of the Pearson skewness coefficient to see how right-skewed the distribution is. If a the distribution is right-skewed, that means there are few players in the top tiers of that position, and the fantasy player might miss out if they don't grab that position this round. If it's left-skewed, that means the top tier of players is quite large, and the AI might want to prioritize other positions.

The Pearson right-skewness coefficient, the fantasy player's ideal roster distribution, and the fantasy player's desired playstyle all go into calculating the position metric that indicates how much the AI wants to draft each position. The AI then transforms all projected fantasy points to z-scores (grouped by position), so that it can compare performance between different positions. These z-scores are then multiplied by the position metric calculated earlier to get a player's final draft value. Once this is done, the AI simply suggests the top 3 players to draft based on draft value. This entire process is done every time it is the player's turn to draft a player.

3.3 Lineup Recommendations (Week to Week Prediction)

In addition to drafting, we also wanted the AI to be able to recommend who to start in a given week. The AI has the choice of any players on the fantasy player's starting roster and bench, as well as any free agents who do not belong to any other fantasy player's team. The AI looks at the matchup of every available football player, comparing their team's previous season's offensive and defensive stats to those of the team said player is facing that week. Using these it then adjusts their projected fantasy points based on the matchup and picks the best available starting roster to recommend. By accounting for the overall offensive ability of a player's NFL team and the defensive ability of their opponent, we are able to predict fantasy points on a per game basis with accuracy far beyond a simple points per game model.

4 Designing a Fantasy Football Application

In addition to making predictions on the season long and week to week performances of NFL players, Gridiron Gurus is a functioning fantasy football application that allows players and to compete against each other and automated bot teams.

4.1 Abandoning YQL and other APIs

Our original application design intended for all management of fantasy football teams to be handled through API calls to 3rd party providers. We looked into both NFL.com and Yahoo.com as potential organizations we could attach Gridiron Gurus to. Unfortunately the official NFL fantasy league no longer supports any of its fantasy APIs and we were not provided with an application key. Additionally, while we were able to register Gridiron Gurus as an application on Yahoo, we ran into problems trying to provide a callback url to Yahoo's authentication API to our application. Because of these issues we decided to make Gridirion Gurus itself a fantasy football application, removing our reliance on these APIs.

4.2 DB Schema

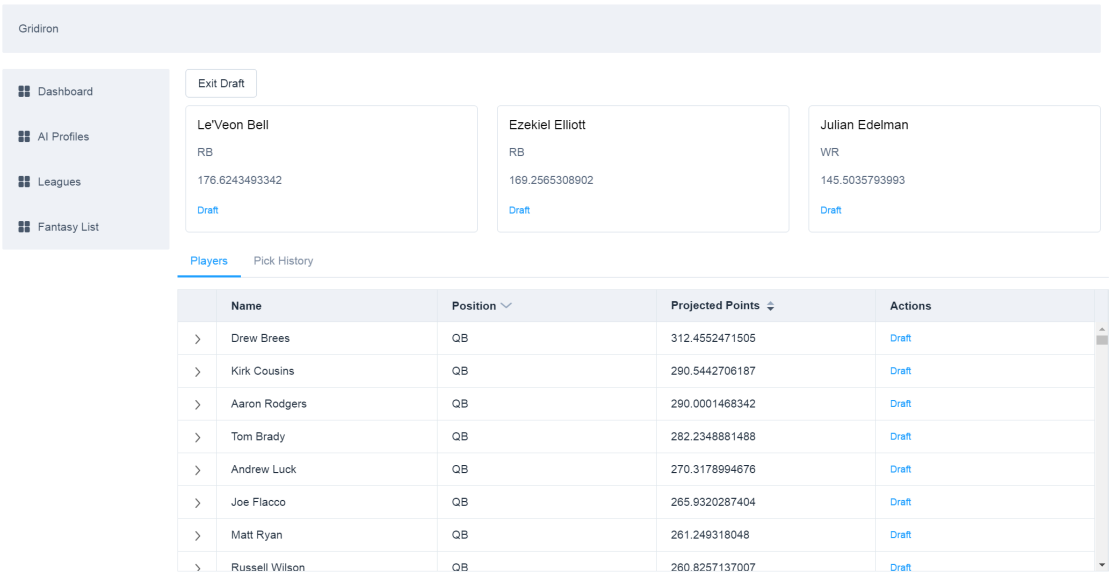
In order to manage the additional fantasy football functionality we had to create a schema comprised of users, fantasy leagues, and teams on our Firebase database. User sign up and sign in are done through Firebase account management calls using an email and password combination. Each user additionally can choose their own non unique screen name. Our database stores users using an auto-generated unique user id that is provided by Firebase on account creation. Leagues are stored in a separate table, with a key that is generated whenever a user decides to create a new fantasy league. Information such as the league name, maximum number of players allowed, and league rules for team composition are requested at league creation and also stored. Additionally, a draft entity is created in another table using the same id as the league it belongs to. Fantasy teams are created at the end of drafts, with references to the players they are comprised of and the user they belong to. A final table keeps track of which team is playing in each league.

4.3 Leagues

Whenever a league is created on our application the commissioner, or league owner, has to set the draft properties before anyone can join the league. These properties include the number of teams in the league, the number of rounds the draft should have, and whether the draft is a snake draft (pick order reverses every round).

Once a league is created it is displayed on the League List and users can join it until it is full. A full league can commence drafting and move to the draft UI. Optionally, a league can start a draft before it is full, and bots will fill the empty spots in the league.

4.4 Drafting



When drafting the user is presented with a list of available players and a list detailing the draft history. For each player predictions for various stats, determined by player position, are displayed. The draft board offers various options for sorting the list of players. At the top of the draft UI Gridiron Gurus provides 3 recommendations on who to draft. These recommendations are calculated using the currently selected AI profile with our season long performance algorithm. At the conclusion of the draft, each team is pushed to the owning player's Team List and associated with the proper leagueId.

By organizing fantasy football leagues between multiple live users we encountered a problem trying to setup a synchronous draft. There wasn't an easy way to allow other users in the league to receive information about your draft pick on your turn. Keeping a person from drafting when it was not their turn was possible, but there was no way for anyone to update their info without refreshing. One way to get around this issue while still showcasing the draft recommendation algorithm was to allow the creation of leagues containing mainly bots and 1 or 0 human players using the optional early draft. By having the AI profiles make their picks immediately after detecting that the user made one, the human player always has the latest draft information. We can still showcase how the recommendation algorithm values positions and picks players given different ideal team compositions and focuses.

4.5 Team List

The Team List displays matchup information for the current active season on each team owned by the user. The user can select any of the teams they have ownership of and view a detailed report for each week in the current season. The report outlines the starting lineup for the team, a prediction of earned fantasy points for your lineup, and the actual results of that week if it has concluded. The report also shows the recommended starting lineup for that team on that week. This recommended lineup is calculated using our week to week prediction algorithm.

5 UI Design

We designed the UI of our application using HTML, and the Vue JavaScript framework. We chose Vue because we saw this as a chance to learn something new and build our programming language repertoire. We used Element, a desktop UI library, as a foundation for our UI choices. This includes forms, tables, buttons, a header and a sidebar. From there we originally tried to build our idea into a website, but once we started dealing with Firebase and the Yahoo auth calls, we realized we had to get around Chrome's security. We moved our idea over to a desktop app, using Electron. Electron allows developers to build cross platform desktop apps with JavaScript, HTML, and CSS.

Vue is the most current and modern framework that the industry is leaning toward. Learning and incorporating Vue into our project was a great way to begin to become fluent in that framework. Vue comes with many nice features, and makes a lot of things that used to be difficult with other frameworks easier.

One of the many benefits of choosing to use Vue is that it has a lightweight virtual DOM (Document Object Model) implementation. This is comparable to React, but Vue's performance is slightly better because of it's DOM. Also, in Vue, a component's dependencies are automatically tracked during render, so the system knows exactly what to re-render when state changes. This allows optimizations to focus more on building the app itself as it scales, rather than the whole class itself. Vue's core library is officially supported and kept up-to-date with the companion libraries for state management and routing. Other frameworks do not officially support or keep these libraries up-to-date. Instead it is left up to the developer, often leading to a fragmented ecosystem.

Vue offers a CLI project generator, which makes it very easy to start a new project using your choice of build system, including webpack, Browserify, or even no build system. Unlike React and other Javascript platforms, Vue offers a variety of templates for various purposes and build systems with no restrictions on how you structure your application. All in all, we wanted a Javascript framework because it makes dynamic view rendering a lot easier.

6 Problems Encountered

At first, we wanted to hook our application up to Yahoo's Fantasy Football API, and have the AI participate in Yahoo's Fantasy Football leagues. However, this slowly became a challenge, as we struggled to make the Yahoo's OAuth work with our application. We tried many things to get the OAuth to grant us a token to access the API with, but after many attempts with no success, we concluded that the problem was not ours, but rather with Yahoo's OAuth system.

We wanted to run our VueJS frontend instance within a web browser. However, due to the default Cross-Origin Resource Sharing policies on modern browsers, such as Chrome or Firefox, the Yahoo OAuth requests would not fire, and throw errors. Because of that, we had to migrate our project from a web application to a standalone Electron application. We managed to find an Electron project template that included Electron bindings for VueJS. Electron runs on the same web engine as the Chrome browser, but it allowed us to disable the web security policies Chrome enforces.

At this point, we were able to have the Yahoo OAuth scripts to run and the login screen show up. Once we login, the OAuth flow requires a redirect back to our own site, passed as a parameter in the URL during the initial request. OAuth must redirect back to localhost, but for some reason, Yahoo would not allow a localhost URL. Doing some research, most APIs support redirects to localhost URLs, but not Yahoo. We tried using a proxy to request Yahoo OAuth access with Auth0, but we later found out it would only work for logins, not API authentication tokens. Finally, we attempted to enter different registered domains in, and those did not work either.

Then, we concluded that Yahoo has a problem with their API. We decided to cut our losses and build our application with its own drafting. It is disappointing when an API lacks any helpful documentation or responsive customer service.

7 Data Sources

All datasets were scraped from their respective sites using the BeautifulSoup python module.

- TheHuddle.com
  - Player and team's statistics by season. The primary dataset used to predict player performance.
- OurLads.com
  - Data of a team's depth chart by season. This indicates who is starting, who is playing backup, etc for every position on a team. The primary dataset used to predict how often a player would play.
- ESPN.com
  - Play-by-play data for a team's offensive statistics. This was used to add extra features to players based on their current season's team's behavior.

8 Work Breakdown

Kyle Tanemura
Michael Li
Erica Dorn
	• Designed and implemented frontend and backend of web application
	  ○ gridiron-electron folder in development branch
	• Set up user authentication and proper page rerouting based on login status
	  ○ gridiron-electron/app/src/renderer/main.js
	  ○ gridiron-electron/app/src/renderer/routes.js
	• Set up remote Firebase database
	  ○ Designed hierarchy to keep track of teams, users, leagues, players, schedules, etc.
	  ○ Establish data access control rules for db security on firebase
	• Set up a system to allow competition between users
	  ○ Joining and creating leagues
	  ○ Scheduling a season between competitors
	• Designed a drafting scheme to allow for multiple users in asynchronous format
	  ○ gridiron-electron/app/src/renderer/pages/Main/Draft.vue

Ryan McKinney	• Predicted player performance using Python
	  ○ Scraped data from online sources
	  ○ All Jupyter Notebooks located in PlayerRanking folder
	• Designed and implemented drafting AI
	  ○ gridiron-electron/app/src/renderer/draft.js
	  ○ gridiron-electron/app/src/renderer/lineup.js