Predicting Fantasy Points Project Report

Jarett Smith, Max Thompson, Charlie Deaton, and Balin Allred

2023-12-14

Contents

Executive Summary

Results:

Our model was able to better predict player performance compared to an industry standard, ESPN. We chose X model that beat ESPN's RMSE by XX. This represents a XX% beat of ESPN's model.

Motivation:

Fantasy sports allows the general public to essentially manage their own virtual franchise. Just like real-world sport franchises, fantasy managers draft their own players according to their belief in a player's ability to attempt to beat other teams in an imaginary "league". According to CNN, roughly 60 million people in North America play fantasy sports. Of the various types of sports, fantasy football is the most popular, with roughly 40 million users. With this large amount of participation, the fantasy sports industry as a whole is worth about $20.3 billion and is expected to continue to grow at 14% annually. This monetary value stems from increased viewership of sporting events and the legalization of sports betting in several countries.

Value-Add:

With this fairly new industry rapidly growing, our goal is to gain a competitive advantage using data mining methods to better predict player performance. By outperforming industry giants like ESPN, there is real opportunity to fiscally benefit for those who are monetarily invested and opportunity to appeal to recreational participants. In short, this model is valuable to both financially invested individuals and to your "just-for-fun", casual participant.

Conclusion:

In summary, our model outperforms ESPN estimates of player performance by XX%. In doing so, we are positioned to provide services to a large audience as higher accuracy predictions have real monetary implications to both die-hard and recreational consumers.

Model	RMSE	RMSE St. Dev.
GLMNet	3.181	0.0234

The Data

Our data comes from play-by-play data of the National Football league dating back to 1999. nflfastR, along with the rest of the nflVerse packages, were used to scrape the data from ProFootballReference, ESPN, and other sites.

At its most granular level, a single row of data represents the performance of a single player, on a single team, for a single week in a single season.

In other words, up to 17 rows of the same player for the same season would make up 1 year of performance for that player.

Player performance

Player performance is measured in fantasy points per game for a single player for a single season. These are generated by the following formula:

```
Fantasy Points = 1 * catches + 0.1 * receiving_yards + 6 * receiving_touchdowns +
  0.1 * rushing_yards + 6 * rushing_touchdowns + 2 * fumbles_lost
```

In other words, pass catchers that catch more passes, for more yards and for more touchdowns will score more fantasy points than those who don't.

In order to account for injuries / incomplete seasons, points per game played is used instead of total points. This is calculated by dividing total points by the number of games played.

Feature Engineering

In order to convert the data from game-by-game to season-by-season, the data was grouped by player and season and aggregated for multiple different features that would later be used in the modeling process.

As a whole, there were four primary types of features that answer the following questions:

1. Player Performance – How good did this player do last year?
2. Player Attributes – What is this player's overall physical and historical background?
3. Team Performance – What is this player's surrounding team like?
4. Player Situation Attributes – What surrounding pieces around this player changed since last year?

Player Performance

In order to predict the performance of player X for season 2022, knowing any information about 2022 performance would be 'cheating' the model, so data was lagged by 1 season to account for this. If one was to predict Tyreek Hill's 2022 points per game, they would only have information on his 2021 performance, not his 2022 performance.

Player Attributes

For a given year, there are attributes of a player that have nothing to do with past performance, but could still be relevant. Some examples include: * years of experience entering this given season * age entering this given season * when the player was drafted (if at all) * an assigned cluster based on physical attributes such as height, weight, and speed

Team Performance

If we know what team a player is starting a given season playing for, we can also know "past performance" of that team. This is done by aggregating the performance of all players on that team for the previous season. Some examples include: * total fantasy points scored by that offense last season * total passing yards / touchdowns scored by that offense last season * information on how good / bad the quarterback on that team played last season

Player Situation Attributes

Finally, even if we know a player's attributes, their past performance, and how their team did last season, there are still changes season-to-season that could affect a player's performance. Some examples include: * did the player change teams? * did this team add / lose any key players? * did this team change their head coach? * did this team invest in players that play the same position during the NFL draft over the offseason?

Final Features:

After considering all four categories, the following features were engineered to use in the modeling process:

Table 2: Fields used for Models

points_pg_ly	years_pro	is_returning_coach
targets_pg_ly	targets_added_this_year	hc_years_with_team
wopr_pg_ly	is_on_new_team	catch_rate_ly
pick	points_per_snap_ly	position
air_yards_pg_ly	targets_per_snap_ly	fp_dropoff
total_games_ly	wopr_per_snap_ly	starter_epa_passing_ly
pass_attempt_difference	total_snaps_ly	starter_epa_persnap_passing_ly
total_positional_investment	total_passing_tds_ly	combine_cluster
target_dropoff	total_passing_yds_ly	qbr_ly_bin
epa_pg_ly	total_passing_fp_ly	

Models

Data Setup

```
master <- filter(master, season >= 2014)
master$pick <- cut(master$pick,breaks = seq(0,400,by=32))
master$pick <- ifelse(master$pick == "(288,320]","Undrafted",master$pick)

current_season$pick <- cut(current_season$pick,breaks = seq(0,400,by=32))
current_season$pick <- ifelse(current_season$pick == "(288,320]","Undrafted",current_season$pick)
```

Train, Test, CV

```r
set.seed(2023)
train_rows <- sample(nrow(master),round(nrow(master)*0.7,1),replace = FALSE)

train <- master[train_rows,]
test <- master[-train_rows,]

ctrl <- trainControl(method = "cv", number = 5)
```

This sets up train and holdout (or test) data sets to train models on. By using train and holdout data sets, we can ensure that our model is not over-fitting to the training data and examine how well the model generalizes to new data.

GLMNet Model

What GLMNet models are and how they work:

GLMNet models use regularized regression to control the complexity of the model and prevent over fitting. These models are particularly useful for high-dimensional data sets where the number of predictors is large relative to the number of observations. The models use traditional linear regression in addition to penalty terms, alpha and lambda, to perform variable selection and handle correlated predictors. The tuning parameter, lambda, controls the strength of regularization, with larger lambda values leading to simpler models. Similarly, larger alpha levels emphasizes lasso regression which also leads to simpler models.

Optimal alpha and lambda levels were found using 5-fold cross-validation. We found the optimal tuning parameters shown below. Essentially, these are the tuning parameters that minimized the RMSE of the model. Given these parameters in our model, the RMSE on the training data set, testing data set, and for 2023 data are shown below:

```r
GLM$results[rownames(GLM$bestTune),c(1:3,6)]
```

```
##    alpha lambda    RMSE     RMSESD
## 77     1      0 3.17549 0.08557425
```

```r
# Train RMSE
TrainResults<- round(postResample(predict(GLM, newdata = train), train$points_per_game)[1], 3)
# Test RMSE
TestResults<- round(postResample(predict(GLM, newdata = test), test$points_per_game)[1], 3)
# 2023 RMSE
Y2023Results<- round(postResample(predict(GLM, newdata = current_season)
                                  , current_season$points_per_game)[1], 3)
```

	RMSE
TrainResults	3.101
TestResults	3.233
Y2023Results	3.406

Now that we had established the optimal parameters, we would re-train the model on all available data except for 2023 data. Using the optimal tuning parameters and training on the full data set, we saw improved performance in both the test data set and the 2023 data set.

```r
GLM_FULL$results[rownames(GLM_FULL$bestTune),c(1:3,6)]
```

```
##   alpha lambda     RMSE     RMSESD
## 1     1      0 3.181193 0.02338361
```

```r
# Master RMSE
MasterResults<- round(postResample(predict(GLM_FULL, newdata = master), master$points_per_game)[1], 3)
# 2023 RMSE
Y2023Results<- round(postResample(predict(GLM_FULL, newdata = current_season)
                                  , current_season$points_per_game)[1], 3)
```

	RMSE
MasterResults	3.125
Y2023Results	3.366

Appendix:

Libraries Used

```
library(dplyr)
library(xgboost)
library(DALEX)
library(glmnet)
library(caret)
library(gbm)
library(tidyverse)
library(nflverse)
library(zoo)
library(plotly)
library(visdat)
library(groupdata2)
library(ggplot2)
library(knitr)
```

Code to Produce GLMNet Model

This code allowed us to choose the best levels for alpha and lambda, which are the two parameters for regualarized regression:

```
alpha <- seq(0,1,by=0.1)
lambda <- seq(-3,0,by=0.5)

set.seed(2023); GLM <- train(points_per_game ~ points_pg_ly + targets_pg_ly +
                        wopr_pg_ly + pick + air_yards_pg_ly + total_games_ly +
                        pass_attempt_difference  + total_positional_investment +
                        target_dropoff + epa_pg_ly + years_pro + targets_added_this_year +
                        is_on_new_team + points_per_snap_ly + targets_per_snap_ly +
                        wopr_per_snap_ly + total_snaps_ly  + total_passing_tds_ly +
                        total_passing_yds_ly + total_passing_fp_ly + is_returning_coach +
                        hc_years_with_team + catch_rate_ly + position + fp_dropoff +
                        starter_epa_passing_ly + starter_epa_persnap_passing_ly +
                        combine_cluster + qbr_ly_bin,
                      data=train,
                      method="glmnet",
                      tuneGrid=expand.grid(alpha = alpha, lambda = lambda),
                      metric="RMSE",
                      verbose=FALSE,
                      trControl=ctrl)
```

After identifying the ideal levels for alpha and lambda via cross-validation, we re-tuned the model to find the ideal coefficients. The new model seen below was tested on all historical data, as well as 2023 data.

```
set.seed(2023); GLM_FULL <- train(points_per_game ~ points_pg_ly + targets_pg_ly +
                             wopr_pg_ly + pick + air_yards_pg_ly + total_games_ly +
                             pass_attempt_difference  + total_positional_investment +
                             target_dropoff + epa_pg_ly + years_pro + targets_added_this_year +
```

```
                    is_on_new_team + points_per_snap_ly + targets_per_snap_ly +
                    wopr_per_snap_ly + total_snaps_ly  + total_passing_tds_ly +
                    total_passing_yds_ly + total_passing_fp_ly + is_returning_coach +
                    hc_years_with_team + catch_rate_ly + position + fp_dropoff +
                    starter_epa_passing_ly + starter_epa_persnap_passing_ly +
                    combine_cluster + qbr_ly_bin,
                 data=master,
                 method="glmnet",
                 tuneGrid=expand.grid(GLM$bestTune),
                 metric="RMSE",
                 verbose=FALSE,
                 trControl=ctrl)
```