

BLOCK POSTERS

Your settings for this Block Poster are:

Pages Wide

Orientation

Paper Format

Border Setting

4

PORTRAIT

LETTER

Top: 0, Right: 0, Bottom: 0, Left: 0

You can find tips on printing, assembling and putting up your poster at www.blockposters.com

Enjoy your Block Poster!

Predicting FanDuel Fantasy Football Production

Problem

We want to predict weekly fantasy football production for all different position groups. Given a player and a week of NFL stats, we feature the player's history and power up requests in regression, we predict the actual fantasy and weekend of production for the player for that week (10-5 seasons).

We built models for all fantasy position groups (QB, RB, WR, TE, DEF, and KICK) using different regression algorithms (Logistic Regression, Gradient Boosting Trees, and Logistic Linear Regression) and different feature sets and classification. Ultimately, we leveraged these predictions for our C2321 project to create optimal line-ups as effectively as can on the FanDuel website.

Data

We downloaded our dataset by scraping: box-score statistics for receiving, rushing, kicking, and passing from NFL.com, offensive line, defensive line, and defensive back from Pro Football Reference, and team offense ranking and DVOA from Football Analytics. We analyzed from FanDuel's website historical fantasy production data for each player from 2013-2018 and downloaded the weekly available player line-ups directly from FanDuel.com.

For training, we used data from the 2013-2018 NFL seasons. For testing, we used data from the ongoing 2019 NFL season.

Results

To get the best mean square error on the classification stage, we use an L_2 regularization parameter to control the complexity of classification buckets.

The table shows mean square error for regression and classification stage for each position group for classification.

Position Group	Regression MSE	Classification MSE
QB	0.000000	0.000000
RB	0.000000	0.000000
WR	0.000000	0.000000
TE	0.000000	0.000000
DEF	0.000000	0.000000
KICK	0.000000	0.000000

Model: Random Forest

Random Forest is a machine learning model that combines the results of many individual weak learners, where each learner is a tree built top-down to ensure all leaves point to one class, which is the final output.

$$f(x) = \frac{1}{N} \sum_{n=1}^N f_n(x)$$

Each tree gives different probabilities for each class; to get the decision function, we average the tree and pick the class with the highest probability.

We used option to use 500 trees of max height 20, with most positive groups only using trees of height 10. This allows us to generalize better by reducing the likelihood of overfitting.

Feature Selection

Based on our evaluation, we note that Gradient Boosting and Logistic Regression perform similarly, but get the edge to the regression stage with a higher accuracy score and lower MSE on all position groups.

Position Group	Regression MSE	Classification MSE	Logistic Reg. MSE	Gradient Boosting MSE
QB	0.000000	0.000000	0.000000	0.000000
RB	0.000000	0.000000	0.000000	0.000000
WR	0.000000	0.000000	0.000000	0.000000
TE	0.000000	0.000000	0.000000	0.000000
DEF	0.000000	0.000000	0.000000	0.000000
KICK	0.000000	0.000000	0.000000	0.000000

Future

The next steps additional features we will like to add including top offensive performance and modeling different types of injuries.

More power-ratings, we could use the interdependence between players, we could use constraint matrices to model them, and we can use optimization to generate the line ups.

References

1. Herman, Bencher. "Machine Learning." (2015). 2001.
2. "Machine Learning." Stanford University. (2015). 2001.
3. "Machine Learning." Stanford University. (2015). 2001.
4. "Machine Learning." Stanford University. (2015). 2001.
5. "Machine Learning." Stanford University. (2015). 2001.
6. "Machine Learning." Stanford University. (2015). 2001.
7. "Machine Learning." Stanford University. (2015). 2001.
8. "Machine Learning." Stanford University. (2015). 2001.
9. "Machine Learning." Stanford University. (2015). 2001.
10. "Machine Learning." Stanford University. (2015). 2001.

Predicting FanDuel

Problem

We sought to predict weekly fantasy football production for different position groups. Given a player and a week of the season, we featurize the player and generate two outputs: with regression, we predict the actual points and with classification, we predict the range (i.e. 10-15 points).

We built models for all six fantasy positions (QB, RB, WR, TE, PK, Def), tested three different learning algorithms (Random Forest, Gradient Boosted Trees, and Logistic/Linear Regression) for both regression and classification. Ultimately, we leveraged these predictions for our CS221 project to create optimal line-ups to effectively bet on the FanDuel website..

Features

- **All:**
 - Difference between team and opponent rank and statistics for last X games and home/away boolean
- **QB:**
 - Passing and rushing yds, passing and rushing TDs, completion percentage, sacks, INTs, QB rating, fumbles lost, pass attempts, EP scored, and salary for last

Fantasy Football Pro

Data

We handbuilt our dataset by scraping: box-score statistics for receiving, rushing, kicking, and passing from NFL.com; offensive line, defensive line, team efficiency, team defense, and team offense rankings and DVOA analysis from FootballOutsiders.com; historical fantasy production data for FanDuel from RotoGuru.com; and we downloaded the weekly available player list directly from FanDuel.com.

For training, we used data from the 2011-2015 NFL season. For testing, we used data from the ongoing 2016 season.

Data	
Pos	Trail
QB	194
WR	952
RB	854
TE	605
PK	238
Def	236

Model: Random Forest

Random Forest utilizes a multitude of decision trees. Each tree is built top-down to ensure all leaves belong to a single class using the Gini impurity:

$$I_G(f) = \sum_{i=1}^J f_i(1 - f_i) = \sum_{i=1}^J (f_i - f_i^2) = \sum_{i=1}^J f_i - \sum_{i=1}^J f_i^2 = 1 - \sum_{i=1}^J f_i^2$$

Production

Results

Set Sizes

In	Test
9	421
0	1681
8	1268
2	905
6	354
8	354

NFL seasons.
5 NFL season.

rest

rees, where
point to one

$$f_i^2 = \sum f_i f_i$$



To get the the mean squares error on the classification algs, we use an expectation generated from the classification buckets.

The table uses mean squares error for regression, and the accuracy score for classification.

Based on our evaluations, we note that Gradient Boosting and

Pos
QB
WR
RB
TE
PK
Def

Pos

Alexei Bastidas (alexeib@stanford.edu)
Ingerid Fosli (ifosli@stanford.edu)



Regression - Mean Squared Error

Random Forest		Gradient Boosting		Linear Regression	
Train	Test	Train	Test	Train	Test
47.121437	49.326750	50.386858	62.032432	48.305206	99.701834
35.396886	32.971057	32.882062	37.831552	31.116383	64.818122
26.223301	31.291023	28.022192	38.211607	25.405439	57.522470
19.903092	18.764963	21.753863	20.372867	20.950363	31.463102
19.006974	16.619200	20.551494	17.125919	19.229433	20.589196
35.643216	45.213687	38.934565	31.286147	33.635457	108.959499

Classification - Accuracy

Random Forest		Gradient Boosting		Logistic Regression	
Train	Test	Train	Test	Train	Test

lost, pass attempts, FP scored, and salary for last 6 games.

- **WR:**
 - Receiving yds, receptions, receiving TDs, fumbles scored, and salary for last 6 games.
- **RB:**
 - Rushing yards, rush attempts, receiving yards, receptions, receiving TDs, avg reception, avg rush scored, and salary for last 3 games.
- **TE:**
 - Receiving yds, receptions, receiving TDs, fumbles scored, and salary for last 3 games.
- **PK:**
 - FG made, FG attempted, PAT made, PAT attempted, points scored, FP scored, and salary for last 3 games.
- **Def:**
 - FP scored and salary for last 6 games.

Automated Feature Selection

For each position group, we leveraged SKLearn's SelectPercentile automated feature extractor to optimize features for classification.

We selected the following top percentiles for each position group:
QB - 90% ; WR - 85% ; RB - 85% ; TE - 85% ; PK - 85% ;
Def - 75%

Each tree gives different probabilities for each of the decision function, we average them and choose the one with the highest probability.

We opted to use 500+ trees of max height 20, while position groups only using trees of height 10. This is to generalize better by reducing the likelihood of overfitting.

Discussion

We define 7 potential output classes: (0 : 0-5 pts), (1: 5-10 pts), (2: 10-15 pts), (3: 15-20 pts), (4: 20-25 pts), (5: 25-30 pts), (6: 30+ pts). Our focus is on the probability, but rather on the entire probability distribution.

Our goal is to generate predictions for betting. This is extremely difficult -- the team based nature of football, the interaction between players on both sides of the ball, which we did not capture.

However, while we could not reliably predict the outcome, looking at the whole probability distribution across all players, we found that the players with the highest expected fantasy point production. So while we are not a good job of selecting high performers on which to bet for a profit -- in fact, we've so far lost \$60 :(

$\sum_{i \neq k} \dots$
 class; to get
 pick the class

with most
 this allows us
 of overfitting.

Random Forests perform very
 similarly, but we give the edge to
 Random Forest on account of a
 higher accuracy score and lower
 MSE on $\frac{5}{6}$ position groups.

QB
WR
RB
TE
PK
Def

s), (1: 5-10 pts), (2: 10-15 pts), (3: 15-20 pts) , (4:
 s is less on the class with the highest
 distribution for a given player.

We found that accurate player predictions are
 Football implies a lot of inter-dependence
 n, as evidenced by our accuracy and MSE

e exact number of points a player will score, by
 cross classes, and sorting by expectation, we
 tion tend to be the players with the highest
 accurately predicting points, we do a surprisingly
 to bet. With that being said, we have yet to turn

Fu

The
 inclu
 likel

Mor
 betw
 mod
 line

R

- 1.
- 2.
- 3.
- 4.
- 5.

0.312821	0.268409	0.328205	0.273159	0.264103	0.182898
0.53729	0.550863	0.532356	0.561570	0.542973	0.505651
0.611696	0.608044	0.612865	0.604101	0.592982	0.589905
0.694467	0.709392	0.681055	0.703867	0.696945	0.690608
0.41841	0.457627	0.393305	0.435028	0.393305	0.378531
0.341772	0.432203	0.316456	0.398305	0.324435	0.302260

uture

There are some additional features we'd like to add, including snap-counts information and modeling likelihood of injuries.

More interestingly, we'd look at the interdependence between players. We could use covariance matrices to model them, and use convex optimization to generate the cups.

References

Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.
 Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics, Vol. 29, No. 5, 2001.
 Friedman, Stochastic Gradient Boosting, 1999
 T. Hastie, R. Tibshirani and J. Friedman. Elements of Statistical Learning Ed. 2, Springer, 2009.
 "Decision Tree Learning", Wikipedia,
https://en.wikipedia.org/wiki/Decision_tree_learning





