Predicting Fantasy Football Production for FanDuel

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

Introduction

Over the last decade, the popularity of fantasy football has risen to the point that multiple betting sites have propped up where users can set a lineup and compete against other people with the promise of cash prizes. The goal of our project is to create models to predict output for all fantasy football position groups - running back, quarterback, wide receiver, tight end, place kicker, and defenses - available on the FanDuel fantasy league betting site. As such, the input is a week in the NFL season, and the output is predictions for all fantasy football position groups for that week. With said models, the ultimate desire is to gain a competitive edge to consistently turn a profit from betting.

Related Work

There has been some work done on fantasy sports, but not academic work. Instead, it's online tools or projects in classes like this one. Most of the work done in Fantasy Football focuses on predicting fantasy points per player, which we also attempt to do. There are multiple methods used; Bayesian inference is used to make predictions for the current season at bayesff.com, initially based on the previous season, but updated throughout. Other lineup generators include fantasyomatic, which uses regression algorithms, and fantasy football analytics, who wrote their own open source R package to scrape online predictions. We haven't found any other predictions based on classification, however.

The regression algorithms used by fantasyomatic is based on a concept they call "Strength of Schedule." The concept can be summarized as the "belief that a player will do better than average against a defense that is weak against their position and worse than average against a defense that is strong against their position" (FantasyOmatic). In other words, to estimate how well a player will do, they focus a lot on how good the defense of the opposing team is against their position.

Dataset Collection

For this project, we built our own dataset by leveraging Python's BeautifulSoup and Urllib packages to scrape data from: FootballOutsiders(FO), RotoGuru(RG), NFL Statistics(NFL), and FanDuel(FD) from 2011 to the current NFL season (2016).

From FO we have been collecting their "revolutionary statistics that break down every single play of the NFL season"- namely Defense-adjusted Value Over Average (DVOA) statistics along with their Defense-adjusted Yards Above Replacement (DYAR) statistic. These statistics are drawn up by FO analysts after reviewing each play of each game using coaches' film and assigned grades to individual players and performances, then computed to normalize across teams so as to get a more informed sense of a team's performance relative to the entire league.

From RG we have been collecting historical FanDuel data - namely, fantasy points achieved, player cost on FanDuel, team a player was on, and their opponent.

From the NFL we have been collecting historical box score information - the number of touchdown passes a quarterback threw in a particular game, the number of yards a running back rushed for, etc.

We've collected this data and loaded it into our own custom data-structure (a series of nested Python dictionaries) in order to quickly generate training samples and prediction data with varying 'gameleads' - the number of game statistics to combine into a feature vector in order to run predictions - along with labels for both regression and classification tasks.

We allocated the 2011-2015 NFL season data for training and k-fold cross validation, and utilized the ongoing 2016 NFL season data as our test set.

Approaches

Feature Selection

Depending on the learning algorithm used, along with the task being modeled (classification vs. regression), we utilized varying game-leads and automated feature selectors. The raw features used include whether the game was played at their home stadium or on the road, fantasy points gained and FanDuel salary for previous weeks, along with team vs opponent aggregate statistics and individual box score statistics.

Gameleads

We utilized varying game-leads for different position groups. By game-lead we mean the number of games worth of statistics to encode into the feature vector in order to predict the upcoming week's production. We found that for Quarterbacks a game-lead of 13 performed best -- which intuitively makes sense as the quarterback position is very mental and really depends on consistently performing well against varying opponents. For other positions we found game leads of 2-6 to work best -- which once again makes intuitive sense as individual skill positions are more centered around weekly match-ups and individual athletic ability, and players in these positions tend to get 'hot' and go on productive streaks.

Team vs Opponent Aggregate Statistics

We use DVOA and DYAR statistics for player's team minus DVOA and DYAR statistics for opponent. Namely, the difference between defensive and offensive rankings as a whole and divided into rushing and passing, the difference between run/pass blocking effectiveness and run/pass defending effectiveness, the difference between sacks given up by offense and sacks gained by defense, the difference in win-loss percentage, and the difference strength of schedule rank. Originally we attempted to input both a player's team and opponent's statistics separately, instead of taking the difference, but our predictions would always pick players from top teams instead of players with the best match-up. By taking the difference we can more accurately model the difference in strength between two teams.

Individual Box Score Statistics

We consider individual statistics tied to positional performance. For example, for quarterbacks we look passing yards, rushing yards, passing touchdowns, rushing touchdowns, interceptions, fumbles, sacks taken, pass completions, quarterback rating, and completion percentage.

Automated Feature Selection - Classification

When fitting our classification models, we leverage Mutual Information to handle automated feature selection by percentile. We found that selecting 70-85% of top correlated features performed best.

Task Definition

Regression

When defining the task that our problem was best suited for, our initial intuition was to fit a regression model -- since we were trying to predict a continuous value. However, while regression yielded decent results, we found that the predicted value gave us little insight into the strength of our prediction.

Classification

With classification, rather than predicting a singular value, we utilized seven discrete intervals for output (0-5, 5-10, 10-15, 15-20, 20-25, 25-30, and 30+ points). We could then feed the entire probability distribution to a lineup generator. To evaluate the results individually, we looked at two interpretations of the output: the predicted class based on max likelihood and the expectation of the output taken over all 7 intervals.

Model Selection

Linear and Logistic Regression

We opted to test linear regression for the regression task and logistic regression for the classification task in efforts to keep the model simple and easy to train. For linear regression we utilized SKLearn's implementation of ordinary least squares with an intercept added. For logistic regression we utilized SKLearn's multi-nomial implementation leveraging a Limited-Memory Broyden-Fletcher-Goldfarb-Shanno solver on account of the relatively low dimensionality of our features. We correctly anticipated that these methods would not perform the best, but tested them to establish a baseline.

Decision Tree Learning

Based on our own prior experience, as well Michael Zhu's advice, we opted to utilize Decision Tree Bagging methods -- namely SKLearn's implementation of Random Forest and Gradient Boosted Trees. Decision Tree Learning algorithms build up individual decision trees by looking at J subsamples of the data in order to create multiple indicators that then each get a 'vote' towards the final prediction. Note that in our best model, Random Forest, the individual trees are built up such that we attempt to minimize the Gini impurity for the entire set. In our code, J is 7:

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

In plain english, the Gini impurity measures how likely a randomly chosen item from the input space would be incorrectly labeled according to the subsampled distribution of labels -- namely, we build the trees up such that we minimize the likelihood of false labels subject to the subsample used by each tree in the forest.
For both regression and classification, we leveraged a large number of shallow trees (500+ trees of max depth 10-20) in order to prevent overfitting, and allow for greater generalization.

Evaluation Metrics

Regression

In order to measure the effectiveness of our regression models, we utilized Mean Squared Error, Mean Absolute Error, and Explained Variance (normalized R^2 coefficients).

Classification

In order to measure the effectiveness of our classification models, we looked at confusion matrices, precision, recall, and f-score measures for each class, and the overall accuracy score. Due to most skill positions (WR, RB, TE, PK) having multiple players per team, many of which don't even play, we found that our datasets tended to be very imbalanced towards having a lot of players that scored 0 points. As such, we did not optimize for accuracy but rather trying to increase precision and recall on non 0 point classes.

Cross-Task Evaluation

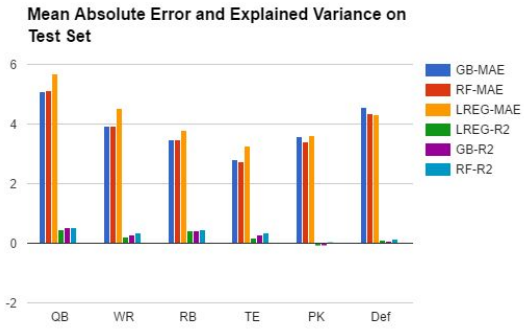
In order to have a way to compare results between the classification models and the regression models, we took the expectation of the predicted probability output of our classification model. In order to do this, we utilized the median as the value for each interval. After taking the expectation, we treat the result as a regressor and are able to compare the Mean Square Error of the varying algorithms.

Results

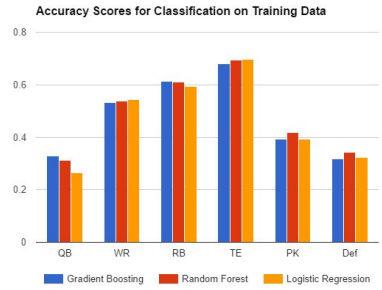
Regression

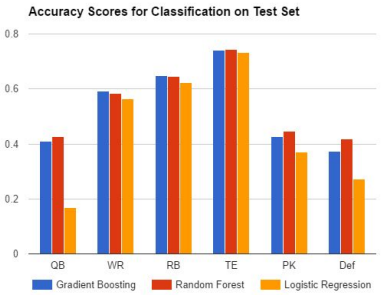
We observed that, after fine-tuning parameters, both decision-tree based algorithms performed similarly in regression: they vary slightly depending on the position group, however, both performed better on the test set than the training set -- indicating good generalization, and decent predictive power. Meanwhile, we note that linear regression performed similarly to the decision tree algorithms on the training set, but did worse on the test set -- indicating that linear regression was quite likely over-fitting to the training data. Nevertheless, across all three algorithms we a mean absolute error of 3-6 points; for our purposes, wherein we're competing against other individuals and winning line-ups can often be separated by fractions of a point, we determine that the predictions are not reliable enough to bet on.

Moreover, we note that Quarterbacks give the highest error rate -- as the most demanding position in football that is most affected by other players, this was expected.



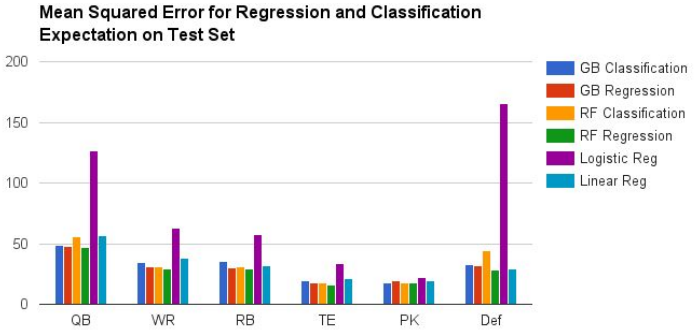
Classification





We observed that, after fine-tuning model parameters, all three algorithms performed similarly on the training data -- with some performing better than others on different position groups. However, on the test data we note that Logistic Regression did significantly worse (particularly with QBs and Defenses), likely due to overfitting. The decision-tree learning algorithms generalized better, with Random Forest edging out Gradient Boosted Trees across all position groups except for WRs. We note, however, that the accuracy scores are skewed on account of the majority of available players scoring a predictable 0-5 points. In fact, across all position groups, if we look at precision and recall for buckets 25-30, and 30+, we find that our average precision is 0.10, and our average recall is 0.05. This hints at the challenge of our task -- properly anticipating players that are not just consistent, but will have 'big' days is difficult as there are a lot of intangibles at play.

Overall



In order to compare the different types of models, we looked at the expectation of the predicted classification probabilities and compared it to the output of the regressors by analyzing mean squared error. We clearly see that logistic regression performed the worst. However, we were surprised to find that the expectation for decision tree based learners performed similarly to their corresponding regressors. This led to our conclusion that the classification learners were indeed better suited for our task: they perform similarly to the regressors, but have the added advantage of providing a probabilistic look at our predictions.

Discussion

To see how useful our models are, we consider how well we estimated the player rankings instead of simply checking how well we predicted each player. To do so, we compare our predicted player rankings to the expert rankings at

QB NFL Rank vs RF classification prediction

week	num in top 5	rank of our first	our rank of first
10	3	3	12
11	4	3	5
12	3	5	2
13	3	4	3
14	1	10	6

QB NFL Rank vs RF regression prediction

week	num in top 5	rank of our first	our rank of first
10	3	3	13
11	3	3	16
12	2	3	14
13	2	1	1
14	2	13	4

http://fantasy.nfl.com/research/rankings. For classification, our highest ranked player is usually in the top 10 and often in the top 5. Sometimes, however, some players have an unexpectedly good or bad week, and our predicted rankings will be way off. For example, see table comparing our predictions for quarterbacks using random forest to the NFL ranking. In regression, our predictions are usually either very close or quite far off, showing that regression doesn't take the variability of players into account as well.

This highlights the challenge of fantasy football modelling -- by predicting an individual player's production, we are essentially trying to predict: the player's team's strategy, the opponent's strategy, individual athleticism, likelihood that a player will be highlighted in the strategy, flow of the game (if a team gets behind it is very likely they will turn to passing more instead of running the ball), likelihood of injury, and a myriad other variables that affect a player's production. We attempted to model strategies and game flow by leveraging FO's DVOA/DYAR rankings -- the idea being if a team has an awful run defense, they will likely be attacked there; individual athleticism/ability through box-score statistics, but we did not account for injury, weather, time of day, practices attended, and countless other factors that affect play.

In turn, we can assert that while our models do not reliably predict the exact production, our classification models are currently helpful for choosing players worth betting on (by virtue of rankings). However, we ultimately will have to revisit our features in order to improve results.

Future Work

Despite our results being far from perfect, they are encouraging and we believe, with additional work, these models could be further tuned to provide reliable betting results. In particular, we w\uld like to incorporate additional features such as time of day, snap-counts (how much a player is on the field during a match), weather, and depth chart placement. We'd also like to include injury modelling to assess: how likely is a player to become injured during a game, and if they become injured, or were previously injured, how much will an injury affect their production? Similarly, we would like to consider utilizing covariance matrices to establish the relationships between teams and players so as to make more accurate, intuitive predictions -- i.e. if we predict that a Quarterback will have an amazing day, this should benefit the predictions for the offensive players on his roster, and bring down production for the opposing defense (something we don't currently model). Lastly, we'd like to develop techniques to do further post-processing on our classification results to approximately determine a player's floor and ceiling for a week -- thereby providing better insight for the person making a bet.

References

1. Andersen, Dennis. "The ffanalytics R Package for Fantasy Football Data Analysis". http://fantasyfootballanalytics.net/2016/06/ffanalytics-r-package-fantasy-football-data-analysis.html
2. BeautifulSoup Python Package Library Documentation https://www.crummy.com/software/BeautifulSoup/
3. Beam, David. "Fantasy Football Projection". http://web.stanford.edu/class/cs221/restricted/projects/dbeam/final.pdf
4. Block, Jason. "Gaining the Edge with Fantasy Football Projections". http://web.stanford.edu/class/cs221/restricted/projects/jblock93/final.pdf

5. cmayer, hughec. “Predicting Fantasy Football Output“. http://web.stanford.edu/class/cs221/restricted/projects/hughec/final.pdf

6. FantasyOmatic. “Our Algorithm”. http://www.fantasyomatic.com/?page_id=4312

7. FootBall Outsiders. “Innovative Statics - DVOA”. http://www.footballoutsiders.com/stats/teameff

8. “Gini Impurity” WikiPedia. https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity

9. Hermann, Eric and Ntoso, Adebia. “Machine Learning Applications in Fantasy Basketball”. http://cs229.stanford.edu/proj2015/104_report.pdf

10. J. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine”, The Annals of Statistics, Vol. 29, No. 5, 2001.

11. J. Friedman, “Stochastic Gradient Boosting”, 1999

12. L. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001.

13. NFL Statistics. http://www.nfl.com/stats/player?seasonId=2016&seasonType=REG&Submit=Go

14. NumPy & SciPy Library Documentation. https://docs.scipy.org/doc/

15. RotoGuru. “FanDuel Historical Data”. http://rotoguru1.com/cgi-bin/fyday.pl?gameyr=fd2011

16. SKLearn Library Documentation. http://scikit-learn.org/stable/documentation.html

17. SKLearn Library Documentation. “Gradient Boosting Classifier”. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

18. SKLearn Library Documentation. “Gradient Boosting Regressor”. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

19. SKLearn Library Documentation. “Linear Regression”. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

20. SKLearn Library Documentation. “Logistic Regression”. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

21. SKLearn Library Documentation. “Random Forest Classifier”. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

22. SKLearn Library Documentation. “Random Forest Regressor”. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

23. T. Hastie, R. Tibshirani and J. Friedman. “Elements of Statistical Learning” Ed. 2, Springer, 2009.