Problem Statement

Title:

Predicting player performance and maximizing team performance in Fantasy Football

Problem:

The inspiration for this project was work done by a Stanford University Computer Science student here. Daily Fantasy Football websites allow users to select a new lineup of NFL players each week to compete against other fans. Each user has a fixed budget for their team, with players with high expected value drawing a high price tag. Users' teams then accrue points based on player performance. The goal of this project is to develop a regression algorithm to predict weekly performance for the players, then determine opportunities where predicted performance is significantly higher than cost to acquire the player. Likewise, I hope to provide an optimization algorithm that, given the fixed budget cap for a team, assembles the team with the maximum expected value. The previously cited project focused only on previous performance, while my aim is to expand this work to include more qualitative factors such as opponent, weather, week of season, etc. A successful model could be used by a Fantasy Football fan to outperform their peers.

Data:

The main source of data is a public repo (https://github.com/BurntSushi/nfldb), containing a SQL database dump of raw NFL game data, dating back to 2009. Of this database, I utilized four tables: plays, teams, games, and players.

I also merged in other data that may have an effect on player performance, specifically weather data(http://www.nflweather.com/) scraped with BeautifulSoup.

Methodology:

I began by merging and flattening the full data set to produce a wide table where each row contains the 'fantasy score' (i.e. performance value) for a given player-game pair, with player and game attributes, such as opponent, age.. Next, I used recent historical fantasy scores to engineer features such as rolling average score and scoring variance. Also, I'll look to non-numeric columns to find certain player affinities, such as correlation between opponent and performance, as well as creating features from other data sources as outlined in the **Data** section.

I'll apply a regression algorithm to predict the weekly performance for every offensive player. Given the data set covers the entire season from multiple years, I'll experiment with using previous years as training data, then using current year for test data as well as using first xx% of the season as training data, then testing the algorithm on the remainder of the season.

Project Goal:

The goal of this project is to provide a standalone repository with which others may replicate and use my prediction algorithm to help guide their own fantasy football season. Specifically, I will include a brief deck outlining my methodology, results, and suggestions for future improvement.

Secondly, I will include the Jupyter notebooks (with instructions) used to wrangle and explore the data, as well as the code for the prediction algorithm.

Project Goal Update:
Having attempted to apply and tune several different regression models to the data, with consistently high error, it's necessary to update the project's goal. Initially, I set out to accurately predict the fantasy for a given player, based on past behavior and other game metadata. Going forward, my goal will be to separate players, teams, and / or games into useful groups, using classification algorithms. Examples of these clusters could be 'low-scoring game overall' or 'advantageous to RBs'. Furthermore, I'll attempt to support or debunk common Fantasy Football techniques, such as never selecting a player in a Thursday game.

Data Wrangling

1. Cleaning
    1. Aggregations & Joins - The raw data included 4 tables (objects), each with at least one unique key, which could be used to join to the other tables. The tables are:
        1. Plays - what were the statistics for a given play and who participated in the play.
        2. Games - what were the total play statistics for the entire game and which teams participated in the game.
        3. Players - what are the summary attributes for a given player.
        4. Teams - Only contains full name, abbreviation, and home city for each team. Will be useful in joining to other data sets later.

        I used GroupBy (by=player,game) to aggregate play-level data to game-player level data. Next I merged player- and game-level data into the previous game-player DataFrame. The result is aggregate statistics for each player, for each game, with metadata from the player (e.g. position) and the game (e.g. home team, away team) tables.

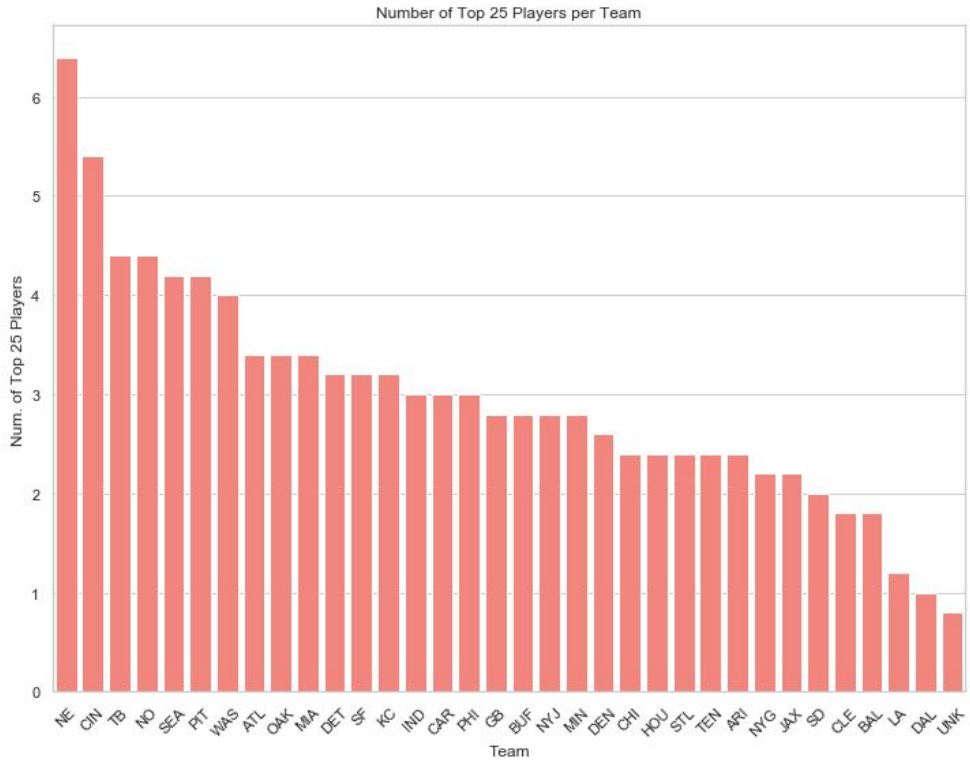
    2. Incomplete Data - There were four columns for name, inconsistently filled throughout the data. I wrote a function to check the columns in order, return the first value found, or combine columns if necessary, and write to a new name column.
    3. Removing Irrelevant data - I removed rows for defensive players, which is not in scope for this project, and removed columns that either have no impact on players' scores or will not be used in features.
    4. Parsing non-numeric data - Weather data provided the temperature and a description of the weather. With something like 65 unique weather descriptions, and no clear pattern for RegEx, I manually categorized the descriptions into Good

Weather, Wet Weather, Windy (but dry) Weather, and Winter Weather, and engineered a new column: desc_simple.

2. Missing Values
    1. Missing weather data - ~16% of the games were missing weather data, with all of 2010 missing weather data. With that outlier removed, 6% at most of any other season is missing data. Later on, I may have to remove 2010 from the model or remove weather as a feature. Filling with the mean does not seem appropriate in this case.
    2. Bye weeks - Each player will have one bye week each season where they will not play. Bye weeks are not included in the data set, so there won't be an observation of 0 point performance due to bye weeks. I won't fill the week with a value, but will just skip over it in any rolling calculations.
3. Outliers
    1. Outliers aren't an issue with this project. In fact, finding outliers with abnormally high scores relative to cost would be an ideal outcome for the project. However, this data set contains data for all players in the selected positions, and given the team salary cap constraint, it is highly likely that I will be selecting only from the top 25 players in any position. The bottom performers will likely score 0 points on most weeks, considering that they won't play at all, so I'll remove all players outside of the top 25 per position.

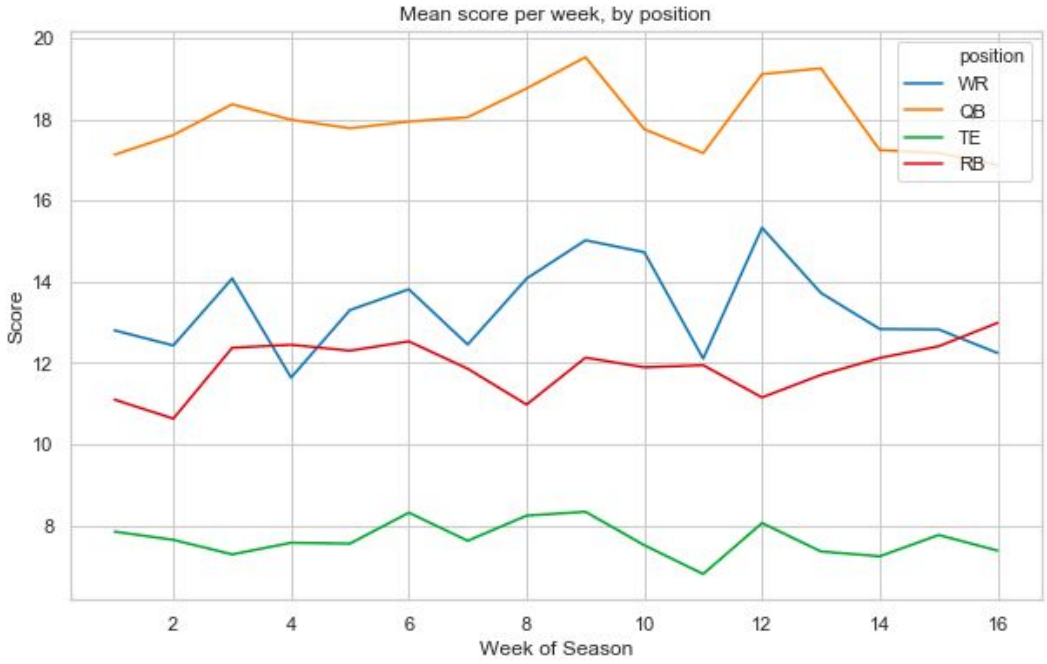
Exploratory Data Analysis

Top Talent Distribution:
The top 25 players at each position over the last 5 years is not evenly distributed amongst all teams. In fact, the most talent dense team, New England Patriots have twice the number of top players than the median and more than 6 times more than the least talent rich team with at least one top 25 player.



Season Fatigue:

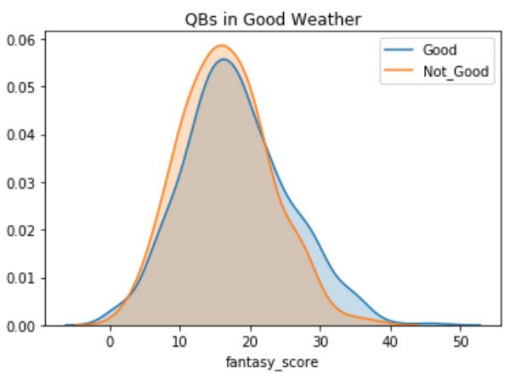
Towards the end of the season, the average score for both WRs and QBs tends to decrease, while RBs actually see an increase in average score in the last few weeks.

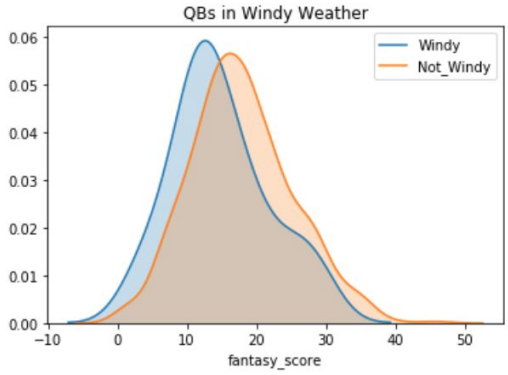


Weather:

Assumption: Quarterbacks can't throw accurately into the wind.
Does windy weather significantly decrease a quarterback's fantasy score? And conversely, does good weather (or playing in a dome) improve performance?
Steps:

1. Filter data set of player-game data to only quarterbacks and split the set into games with `windy` weather vs. non-`windy` weather.
2. Plot distribution of fantasy score for both sets.





3. Perform 2 sample, one-sided z-test, with alpha = 0.05 where:
   $H(0): \mu_{windy} = \mu_{not\ windy}$
   $H(a): \mu_{windy} < \mu_{not\ windy}$
4. Determine significance and repeat with `good` vs. not `good` weather data.
5. We find a significant positive (p = 0.0027) for `good` weather and a significant negative (p = 0.0476) impact of wind on quarterback performance.

Thursday Night Curse:
Assumption: Teams that play on Thursday have had a shorter week to recover and are less likely to play with intensity.
Is the average fantasy score lower than the average score the rest of the week? In terms of actual score (not fantasy points), are Thursday games lower scoring on average?
Steps:

A. Average Fantasy Score
   a. Split player-game data set into Thursday and non-Thursday games.
   b. Perform 2 sample, one-sided z-test, with alpha = 0.05 where:
      $H(0): \mu_{Thurs} = \mu_{non\text{-}Thurs}$
      $H(a): \mu_{Thurs} < \mu_{non\text{-}Thurs}$
   c. Determine significance (p=0.303) and fail to reject the null hypothesis. No significant difference in players' fantasy scores.

B. Average Total Game Score
   a. Compute total game score as home_score + away_score
   b. Subset data into Thursday, non-Thursday

c. Apply bootstrapping by 1) computing difference in means between the two data sets 2) permuting the samples and 3) counting bootstrapped samples where difference of means is at least as extreme as our original samples (p=0.1948).
d. Determine signification - again, no significant difference in total game score between Thurs. and non-Thurs. games.

Share of Targets:
Assumption: Teams force the ball to their best players. Getting more of the targets will yield them a higher fantasy score.
What's the correlation between the relevant 'opportunity' metrics and fantasy score at each position?
Using pearsonr():

Metric	Pearson R (Correlation Coefficient)
QB - Passing Attempts	0.3178
QB - Rushing Attempts	0.2052
RB - Rushing Attempts	0.598
RB - Targets	0.373
WR - Targets	0.5575
TE - Targets	0.6845

There's a moderate positive correlation between opportunity and fantasy score for RBs, WRs, and TEs. Strangely enough, the position handing out the football, the QB, has a low correlation with passes or rushing attempts with fantasy score. This may raise additional questions around efficiency of player e.g. pass completion percentage.

Predictive Modeling

Data Structure
Each row in the data represents an observation; that is a game for a given player. The observation contains numeric features describing previous performance, categorical variables for position, team, day of week, etc., and the target variable, fantasy score. After creating dummies, there are 117 total features.

Regression Approach

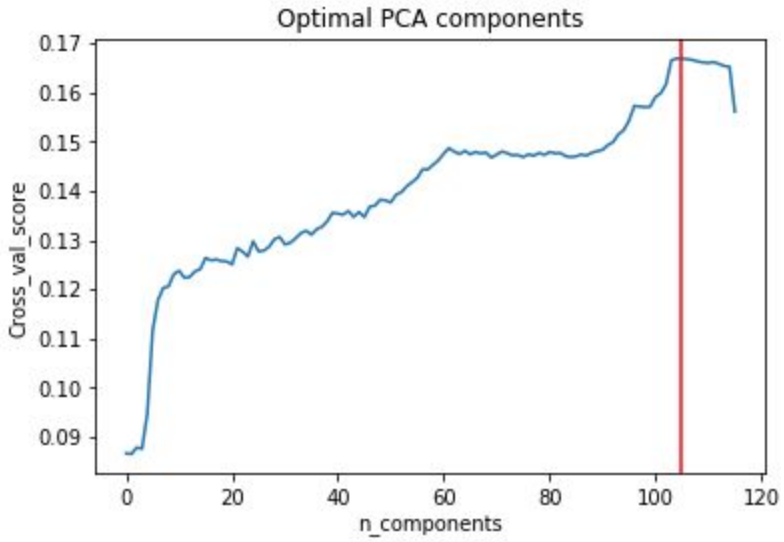
My general approach to finding an appropriate regression algorithm was:

1. Transform the data set into a usable format by sklearn, specifically by using standard scaling, creating dummies for categorical variables and splitting into training and test sets.
2. Apply PCA and determining the optimal number of components.
3. Choose a variety of regression models and tune hyperparameters using 5-fold cross-validation.
4. Record mean-squared-error against test data for each regressor and plot to determine the most appropriate model for this problem.

PCA

Using a Ridge regressor with default parameters to determine optimal n_components for PCA:

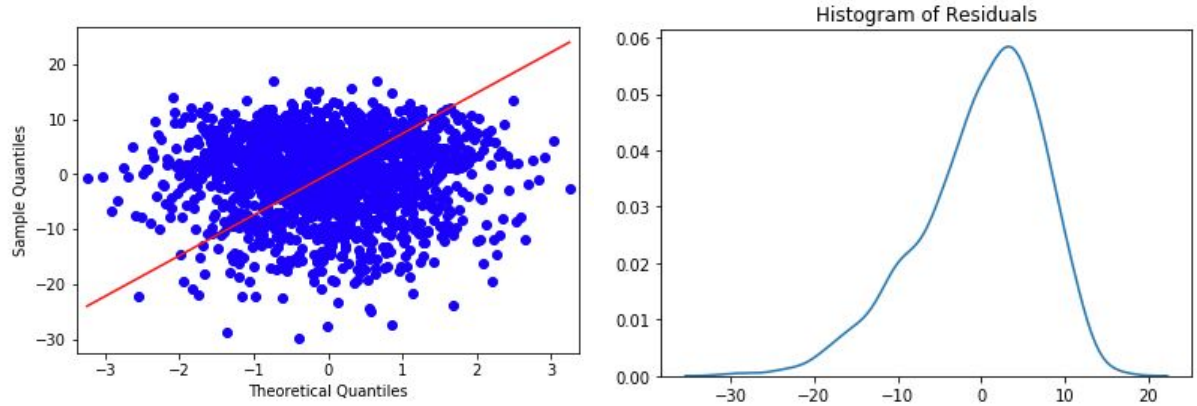
```
n_components: 105
```



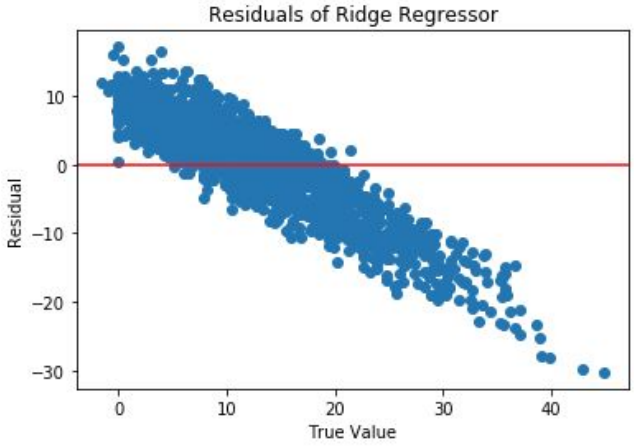
Ridge Regression

Using PCA parameter above, with 5-fold cross-validation and hyperparameter tuning, the regressor resulted in MSE of 54.66, with explained variance score of 0.2198. Given the ease of training and deploying this model, Ridge will be the baseline against which other models will compared.

Residual analysis (QQ plot below) shows no clear pattern, which doesn't lead me to believe a non-linear model is necessary. Likewise, in plotting a histogram, the residuals are approximately normal, with slight left skew. This skew is likely due to the difficulty in predicting 'blowouts', when a player drastically overperforms. In the other direction, there are more safeguards against 'busts', with it being very difficult for a player to score less than zero.

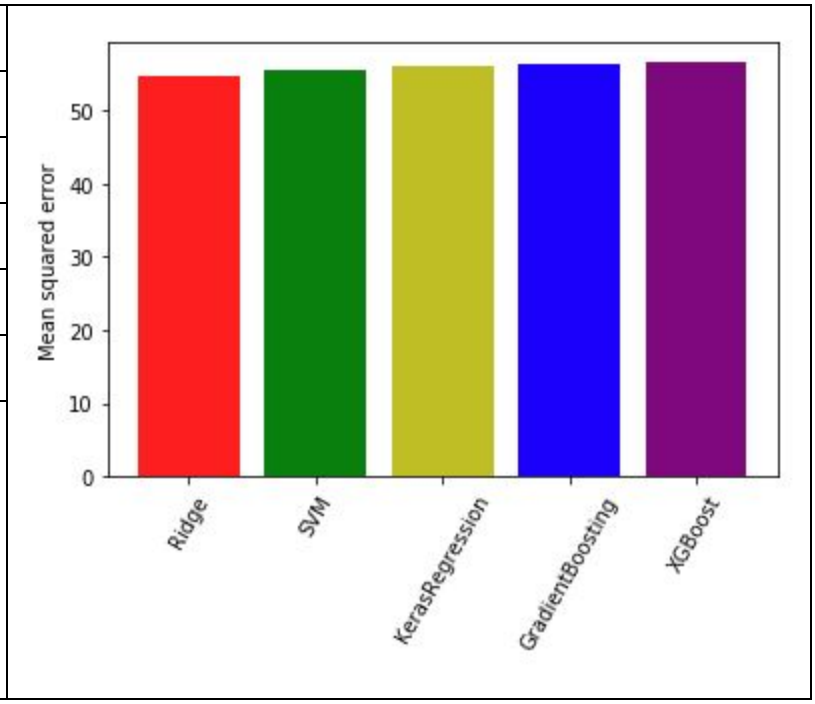


The theory above, that the models can't predict boom-or-bust events well, is displayed in the residual plot below. It's show than the model performs well in a rather wide range of average scores (~8-16), but when top players have terrible or phenomenal games, the predictions break down.



Comparison to other regression models:

Model	MSE
Ridge	54.60
SVM	55.32
KerasRegressor	55.85
GradientBoosting	56.38
XGBoost	56.51



After considering the minimum MSE attained by the best performing model, I found that an expected error range was far too high to be useful in practical application. For example, if the error interval is ± 5, and the average score for a given position is ~15, it's clear that the predictions are not valuable.Due to the large MSE of linear models, I also tried using a neural network, which gave similar error scores. A secondary approach using classifiers to select the optimal player lineup is described in the next section.

Classification Approach

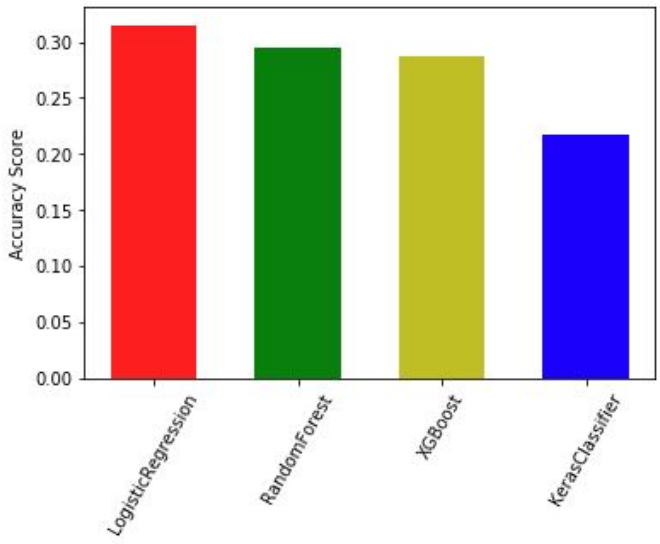
Due to the high variance in player performance, likely due to randomness, regression algorithms were unable to accurately predict outcomes. For more robustness against variance due to randomness, I modified the target variable from a numeric score to a categorical score range (0-5, 5-10, etc.), in five point increments. As a result, I adjusted the problem from regression to classification. Once again, I compared various tuned models to make predictions for score range on the test data set, using accuracy rather than mean-square-error to compare model performance.

LogisticRegression

The baseline model I chose for classification was logistic regression, which resulted in an accuracy of 0.3150. Similar to the regression approach, the simpler model (logistic regression) resulted in the greatest, though still very poor, accuracy.

Comparison to other classification models:

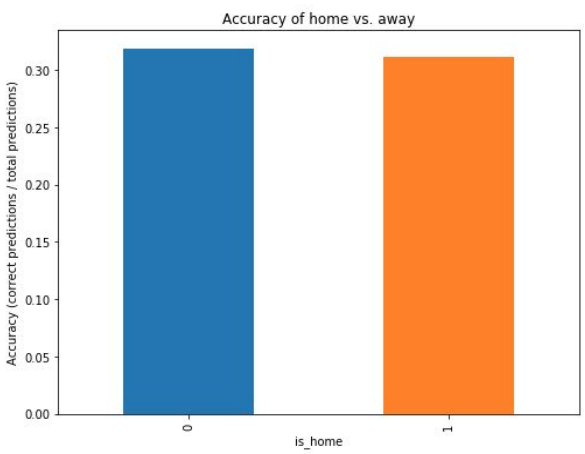
Model	Accuracy
LogisticRegression	0.3150
RandomForestClassifier	0.2943
XGBoost	0.289
KerasClassifier	0.2175

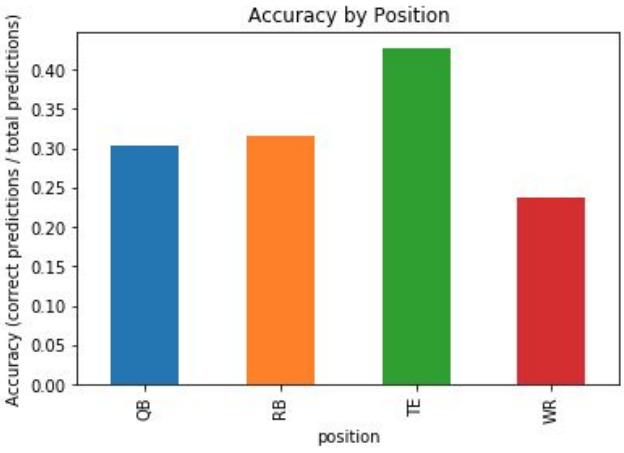


Per-player Neural Network

Given the poor accuracy results from classification and regression models against anonymous predictions, I added `player_id` as feature, which approximately doubled the number of features to 325, once `player_id` was transformed into dummies. Using KerasRegressor again, this yielded a MSE of 55.89, essentially identical to the MSE of the anonymous KerasRegressor.

Classification Accuracy across Subgroups





a) No difference in accuracy in predicting scores for home or away players. This could be because there is no real effect on a player's score or that the model controls for that variance.
b) On the other hand, there is a large difference in accuracy of predictions between positions. Interestingly, TE's have the lowest variance in scores, but RBs have the highest variance.

Conclusions

In looking back to the project goal of accurately predicting the fantasy football performance of players given only known past performance and general information about the upcoming game, predictive modeling did not succeed. Regression algorithms consistently produced margins of error outside of a usable range and classification algorithm placing players into scoring bands were similarly inaccurate. In terms of residual analysis, it appears that the models are not failing due to poor model choice (e.g. nonlinearity), but rather that the selected features do not account well for observations significantly outside the mean, which we see is quite common in the exploratory data analysis section.

For future work, I would attempt to engineer additional features to account for more of the variance, particularly on the player-level and dealing with sequential data. However, it is quite likely that the amount of unexplainable variance, or true randomness, that cannot be accounted for using prior data, is sufficiently high that predictive modelling will continue to be inaccurate.

Fortunately, I was able to glean some insights that benefit the overarching goal of improving fantasy football team selection on any given week. By factoring in weather forecasts into QB selection or considering adding low-cost Thursday players, users can gain a competitive edge over opponents who rely more heavily on popular knowledge than the data.