We sought to predict weekly fantasy football production for six different position groups. Given a player and a week of the NFL season, we featureize 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.

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

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 seasons. For testing, we used data from the ongoing 2016 NFL season.

1. **QB**: Receiving yards, receptions, receiving touchdowns, fumbles lost, fantasy points scored, and salary for last 6 games.
2. **RB**: Rushing yards, rushing attempts, receiving yards, receptions, receiving touchdowns, average reception, average rush, fantasy points scored, and salary for last 6 games.
3. **WR**: Receiving yards, receptions, receiving touchdowns, fumbles lost, fantasy points scored, and salary for last 3 games.
4. **TE**: Receiving yards, receptions, receiving touchdowns, fantasy points scored, and salary for last 3 games.
5. **PK**: FG made, FG attempted, PAT made, PAT attempted, fantasy points scored, and salary for last 3 games.
6. **Def**: Team efficiency, team defense, fantasy points scored, and salary for last 6 games.

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

### Automated Feature Selection:
For each position group, we leveraged SKLearn’s SelectPercentile automated feature extractor to optimize the features for classification.
We selected the following top percentiles for each position:
- **QB** - 90%; **WR** - 85%; **RB** - 85%; **TE** - 85%; **PK** - 85%; **Def** - 75%

### Model: Random Forest
Random Forest utilizes a multitude of decision trees, where each tree is built top-down to ensure all leaves point to one class using the Gini impurity:

$$I_G(f) = \frac{1}{2} \sum_{i=1}^{n} \left( f_i - f_{\text{avg}} \right)^2$$

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

We opted to use 500+ trees of max height 20, with most position groups only using trees of height 10. This allows us 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 less on the class with the highest probability, but rather on the entire probability distribution for a given player.

Our goal is to generate predictions for betting. We found that accurate player predictions are extremely difficult -- the team-based nature of football implies a lot of inter-dependence between players on both sides of the ball, which, as evidenced by our accuracy and MSE scores, we did not capture.

However, while we could not reliably predict the exact number of points a player will score, by looking at the whole probability distribution across classes, and sorting by expectation, we found that the players with the highest expectation tend to be the players with the highest fantasy point production. So while we are not accurately predicting points, we do a surprisingly good job of selecting high performers on which to bet. With that being said, we have yet to turn a profit -- in fact, we've so far lost $60 :(

### References
3. J. Friedman, Stochastic Gradient Boosting, 1999

### Table

<table>
<thead>
<tr>
<th>Position</th>
<th>Features</th>
<th>Dataset Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>QB</td>
<td>Past</td>
<td>Train</td>
</tr>
<tr>
<td>WR</td>
<td>Past</td>
<td>1949</td>
</tr>
<tr>
<td>RB</td>
<td>Past</td>
<td>9520</td>
</tr>
<tr>
<td>TE</td>
<td>Past</td>
<td>8548</td>
</tr>
<tr>
<td>PK</td>
<td>Past</td>
<td>2052</td>
</tr>
<tr>
<td>Def</td>
<td>Past</td>
<td>2366</td>
</tr>
</tbody>
</table>

### Results
To get the the mean squares error on the classification alg, 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 Random Forests perform very similarly, but we give the edge to Random Forest on account of a slightly lower MSE than the Random Forest.