Winning Daily Fantasy Football Contests Using Kernel Methods

Alan Du

Michael Chiang

May 2017

Abstract

Previous literature on Daily Fantasy Football has focused on predicting the number of fantasy points earned by each player in a regression setting. These attempts have yielded mediocre results due to the large amount of noise observed week to week. High regression errors have been a roadblock to developing a winning system. In this paper, we tackle the problem with a classification rather than regression approach—that is, instead of predicting the precise number of fantasy points earned by a player, we predict whether or not the player will "go off". Finding players that score exceedingly higher than expected is one of the keys to winning Daily Fantasy Football contests with top-heavy payout structures. We train various classifiers and find that asymmetric cost SVM performs well—we manage to increase precision from 0% to over 40% in some weeks. We test our models on the "DraftKings NFL $1M Play-Action" contest over the 2016-2017 regular season. Our approach beats the profit of the baseline strategy by 322%.

Contents

1 Background and Motivation **3**

1.1 Daily Fantasy Sports 3

1.2 DraftKings NFL Rules 3

1.3 DraftKings NFL Payout Structure 4

1.4 "Expert" Projection Sources 5

2 Related Work **6**

3 Data **8**

3.1 QBs 9

3.2 RBs 9

3.3 WRs 9

3.4 TEs 10

4 Methodology **11**

4.1 Stage I: Projections 11

4.1.1 Non-Kernel Methods 11

4.1.2 SVM 12

4.1.3 Asymmetric Cost SVM 12

4.1.4 PCA 14

4.2 Stage II: Combinatorial Optimization 14

5 Results **18**

5.1 Spiking WR Projections Using Asymmetric Cost SVM with Linear Kernel 18

5.1.1 Projections 18
5.1.2 Combinatorial Optimization 19

6 Conclusion **24**

6.1 Discussion 24
6.2 Future Work 25

Chapter 1

Background and Motivation

1.1 Daily Fantasy Sports

In Daily Fantasy Sports (DFS) contests, contestants construct a virtual lineup of players that score points based on their real-world performances. Unlike in season-long Fantasy Sports contests,in DFS contestants submit a new lineup for each set of games. DFS contests are held for several professional sports leagues, including the National Football League (NFL), National Basketball League (NBA), and National Hockey League (NHL). The leading DFS sites today are DraftKings and Fanduel, which control approximately 90% of the $3B DFS market.

There are three primary types of DFS games: Head-to-Heads (H2Hs), Double-Ups, and Guaranteed Prize Pools (GPPs). In H2H games, two contestants play for a single cash prize. In Double-Up games, a pool of contestants compete to place in the top 50% of lineups, which are awarded twice the entry fee. In GPPs, a pool of contestants compete for a fixed prize structure that tends to be very top heavy. We choose to play the "NFL $1M Play-Action" GPP contest on DraftKings.

1.2 DraftKings NFL Rules

The full list of rules can be found at https://www.draftkings.com/help/nfl. Here are the highlights:

- Each player listed has an assigned salary and a valid lineup must not exceed the salary cap of $50,000.
- Rosters will consist of 9 players and must include players from at least 2 different NFL teams.
- The 9 roster positions are QB, RB1, RB2, WR1, WR2, WR3, TE, FLEX (RB/WR/TE), and DST.
- Lineups may be edited at any time leading up to games. Each individual player will become "locked" at the scheduled start time of their team's game.

1.3 DraftKings NFL Payout Structure

The most popular GPPs on DraftKings are the "NFL $1M Play-Action" ($3 entry fee) and "NFL $4.44M Fantasy Football Millionaire" ($20 entry fee) contests. The "NFL $1M Play-Action" contest allows for a maximum of 150 entries per contestant and has a total entry limit of about 450,000. Figure 1.1 illustrates the top-heavy payout structure. Typically, the cutoff for cashing lineups is at the 75th percentile, which pays $5 in this contest.

PRIZE PAYOUTS	
1st	$50,000.00
2nd	$25,000.00
3rd	$15,000.00
4th	$10,000.00
5th	$7,500.00
6th	$5,000.00
7th - 8th	$3,000.00
9th - 10th	$2,000.00
11th - 15th	$1,500.00
16th - 25th	$1,000.00
26th - 35th	$750.00
36th - 50th	$500.00

Figure 1.1: "NFL $1M Play-Action" Payout Structure, Top 50 Places

1.4 "Expert" Projection Sources

There are numerous sites that offer projections developed by "experts", such as Daily Fantasy Nerd and RotoGrinders. While many DFS contestants rely on these projections, they are oftentimes inaccurate. Specifically, they err on the conservative, so cheaper players are rarely given high projections. In order to develop a winning system, we need to identify cheaper players that have the potential to go off. Therefore, we seek to use machine learning algorithms to predict which players will score big.

Chapter 2

Related Work

Previous work on improving fantasy point projections has been in the regression setting. Lutz [1] used support vector regression and neural networks to predict the number of fantasy points earned by each player. Focusing on QBs, he concluded that errors were too high after achieving an MAE of over 6 fantasy points using support vector regression (neural networks performed even worse).

Parikh [2] used support vector regression and k-nearest neighbor regression to predict the number of fantasy points earned by each player in every week of the 2014-2015 season. He found that SVM with RBF kernel performed the best across all positions. Relative to a benchmark projection error, the model reduced projection errors for TEs, WRs, and RBs but increased errors for QBs. He did not test the improved projections for TEs, WRs, and RBs by generating lineups and comparing to historical contest results, so we do not know if this attempt at predicting fantasy points would realize profits.

Dunnington [3] evaluated the accuracy of expert projections. He concluded that the projections from four expert sites (NumberFire, Pro Football Focus, 4for4, and BSports) were highly correlated and do not statistically differ in accuracy. He also discovered that information aggregation ("wisdom of the crowds") does not lead to a statistically significant increase in accuracy.

Hunter et al. [4] designed a greedy integer programming formulation that maximizes the probability of at least one lineup winning. The integer program takes in a set of

salaries and fantasy point projections as input and produces a set of N lineups as output. For $i \in \{1, ..., N\}$, the optimal solution maximizes the expected fantasy point score of lineup i, lower bounds the variance of lineup i, and upper bounds the correlation of lineup i with lineup $j \in \{1, ..., i-1\}$. Using a weighted average of RotoGrinders and Daily Fantasy Nerd projections, the authors used the greedy integer programming formulation to win several contests in Daily Fantasy Hockey and Daily Fantasy Baseball.

Chapter 3

Data

We scraped data throughout the 2016-2017 NFL season from various sources. The overall dataset has 3,309 examples and 35 features. Each example represents one player in one game. Since the last three weeks of fantasy points are included in the feature set, our data begins in week 4. Furthermore, we excluded week 17 because the conditions in this week differ from other weeks—many teams choose to give backups more playing time since the outcome of their season has already been decided. Here is a partial list of our feature set:

- Daily Fantasy Nerd and RotoGrinders Predictions
- Salary
- Vegas Spread
- Historical Fantasy Points (Last 3 Weeks)
- Rush Attempts and Targets (Last 3 Weeks)
- Rush Attempts and Targets (Total)
- RedZone Rush Attempts and Targets (Total)
- TDs (Total)
- Opposing Defense Rank
- Opposing Defense Passing Yards Per Game
- Opposing Defense Rushing Yards Per Game

- Projected Rush Attempts and Targets (Daily Fantasy Nerd)
- Projected Floor and Ceiling Fantasy Points (Daily Fantasy Nerd)

We subset the data by position because fantasy points for some positions tend to more easily predicted than others [3]. Since our goal is to predict which cheap players will score highly, we further subset the data using a salary threshold. We also set a fantasy points threshold to define what it means for a player to go off, which we will refer to as an "efficiency threshold". The response variable that we are trying to predict is +1 if the player exceeds the efficiency threshold for his position and -1 otherwise. We provide details on the dataset for each position below.

3.1 QBs

There are 287 cheap QBs using a salary threshold of $6000. With an efficiency threshold of 25 fantasy points, there are 28 players that exceed the efficiency threshold.

3.2 RBs

There are 739 cheap RBs using a salary threshold of $5000. With an efficiency threshold of 18.5 fantasy points, there are 65 players that exceed the efficiency threshold. Since there were very few RBs under a salary of $3500 that earned more than 0.0 fantasy points, we removed these RBs. In the resulting dataset, there are 401 cheap RBs and 54 players that exceed the efficiency threshold.

3.3 WRs

There are 1027 cheap WRs using a salary threshold of $5000. With an efficiency threshold of 18.5 fantasy points, there are 90 players that exceed the efficiency threshold.

3.4 TEs

There are 506 cheap TEs using a salary threshold of $3500. With an efficiency threshold of 16 fantasy points, there are 65 players that exceed the efficiency threshold. Since there were very few TEs with a salary of $2500 or less that earned more than 0.0 fantasy points, we removed these TEs. In the resulting dataset, there are 206 cheap TEs and 29 players that exceed the efficiency threshold.

Chapter 4

Methodology

Our methodology can be broken down into two stages. In Stage I, we improve expert projections using machine learning algorithms. In Stage II, we generate an optimal set of lineups using the new projections for each week and test the lineups in the "NFL $1M Play-Action" contest for the 2016-2017 season.

In order to avoid look-ahead bias, we had to be careful with which data we trained our models with. For example, if we were to use these models in practice, to avoid look-ahead bias we cannot use a model trained on data from weeks 4-16 to predict which players will go off in week 10. In the following sections, we train models using weeks 4-16 to get a general idea of how they would perform in a live setting, and then train them on a rolling basis (i.e. using weeks 4-6 to predict week 7) for testing without look-ahead bias.

4.1 Stage I: Projections

4.1.1 Non-Kernel Methods

First we try various machine learning algorithms that are standard in the classification setting. We train these models using data from weeks 4-16 to get a general idea of how they would perform in a live setting (keeping look-ahead bias in mind). We train the following classifiers for each position:

- Logistic Regression

- Logistic Regression with LASSO Penalty
- Logistic Regression with Ridge Penalty
- Logistic Regression with Elastic-Net Penalty ($\alpha = 0.5$)
- Classification Tree
- Random Forest

Unfortunately, these classifiers are not useful for our purposes. All of them have reasonable accuracy but poor precision—in fact, for WRs, none of the classifiers predict any +1's on the training or testing sets. The reason for the low precision is the sparsity of the response vector. Because so few players exceed the efficiency threshold, these classifiers always predict -1, which does not help us improve predictions.

4.1.2 SVM

The initial kernel method we try is soft-margin SVM . Let us assume that we have n labeled examples $(x_1, y_1), \ldots, (x_n, y_n)$ with labels $y_i \in \{+1, -1\}$. For $w \in R^d$ and $b \in R$, we solve the following optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, n$$

Each label y_i represents whether or not the player exceeds the efficiency threshold, and each x_i is a vector of $d = 35$ features. Unfortunately, just like the classifiers in Section 4.1.1, the SVM approach has poor precision. For WRs, there are again no 1's predicted on the training or testing set.

4.1.3 Asymmetric Cost SVM

The sparsity of the response vector is a challenge that standard classifiers cannot overcome. In order to achieve our goal of predicting which players will exceed the efficiency

threshold, we need our classifier to predict some number of +1's. In fact, we would generally prefer the classifier to predict +1's over -1's because missing out on a player that goes off tends to be more damaging from a profit and loss standpoint than incorrectly predicting that a mediocre player will go off. In other words, the cost of mislabeling a +1 player as -1 is higher than the cost of mislabeling a -1 player as +1. Thus, our sensitivity to costs is asymmetric—we are relatively less sensitive to false-positives and relatively more sensitive to false-negatives.

Using a method first introduced by Morik et al. in the medical field [5], we adjust for our asymmetric cost sensitivity by assigning unequal weights to false-positive and false-negative penalties. We introduce cost factors C_+ and C_- for false-positives and false-negatives, respectively, and solve the following optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C_+ \sum_{i:y_i=+1} \xi_i + C_- \sum_{j:y_j=-1} \xi_j$$

$$\text{s.t. } y_k(w^T x_k + b) \geq 1 - \xi_k,\ k = 1, \ldots, n$$

We use SVM$^{\text{light}}$ [6], a C language implementation of SVMs with asymmetric cost factors capability, to find the optimal solution. We tune the C_+ and C_- hyperparameters using a simple heuristic. First, we compute the following ratio:

$$\frac{C_+}{C_-} = \frac{\text{number of -1 training examples}}{\text{number of +1 training examples}}$$

This ratio sets the potential total cost of false positives equal to the potential total cost of false negatives. Next, we generate a sequence of (C_+, C_-) pairs that are centered around the (C_+, C_-) pair computed using the above ratio. The step size of the (C_+, C_-) sequence is limited by training time; we generally stick with a step size of 0.05. Using 5-fold cross-validation, we solve the optimization problem for each (C_+, C_-) pair. Instead of simply selecting the model with the lowest cross-validation error, we keep the model that achieves the lowest cross-validation error *and* predicts at least 10% of examples to be +1. We train the model using linear and RBF kernels and find that this method is a significant improvement on cost-symmetric SVM and non-kernel methods. It performs

well on the testing set for weeks 4-16 and also performs well in a live weekly testing setting where we remove look-ahead bias. Full details can be found in Chapter 5.

We experimented with various ways of utilizing the predictions. Ultimately, we decided that the best approach is to spike the expert projections of players that the asymmetric cost SVM classifier predicts will go off. Specifically, we spike the Daily Fantasy Nerd projections by a factor of 1.5 for the players that we predict will go off.

4.1.4 PCA

In practice, we could only have around 45 minutes to tune this model and solve the combinatorial optimization problem of selecting lineups. This would happen if the final injury reports, which are confirmed an hour because the contest is closed, showed that the players who we predicted to go off are injured. In this extreme case, we would not be able to tune an asymmetric cost SVM due to its long run time. In our experiments it took a couple hours to tune properly.

We choose to stick with the asymmetric cost SVM because the sparsity problem still exists. However, instead of running the SVM on all our data we choose to run a PCA on our data, select a top subsection of components as a dataset and then tune the SVMs in order to see if we could hold the accuracy of the SVM while cutting run time down dramatically due to a lower dimension of parameters.

4.2 Stage II: Combinatorial Optimization

As we have already discussed, we are intersted in two distinct types of contests: GPPs and Double-Ups. The approach to the combinatorial optimization problem that we will take differs for each different game due to different incentives.

As we have noted in Section 1.3, the payout structure of a GPP is very top heavy with 1st place winning up to 40,000x their entry fee. Since we are allowed to enter 150 lineups into this contest, it is fairly intuitive to approach this problem by maximizing our top lineup score rather than maximizing the lowest score. We do this by borrowing from

portfolio optimization theory and attempting to make each of our 150 assets (lineups) as negatively correlated to one another as possible while also maximizing the correlation between our players in each lineup. We solve this problem with a greedy integer program with various constraints that ensure different types of generated lineups.

Greedy Integer Program

Inspired by Hunter et al., we use a greedy integer program to generate lineups. We use lineup constraints based on analysis done on historical daily fantasy football data to determine correlations and trends in winning lineups. We construct "formulations" with different constraints.

Formulations

This section will discuss some of the different formulations we use and the constraints that define each formulation. All formulations seek to maximize the number of projected points in a lineup.

Formulation 0 - This is our feasibility formulation. The constraints that are made are only in place to construct a DFS lineup that abides by the specific rules stated in Section 1.2.

$$\sum_{k \in QB} x_{ik} = 1$$

$$2 \leq \sum_{k \in RB} x_{ik} \geq 3$$

$$3 \leq \sum_{k \in WR} x_{ik} \geq 4$$

$$1 \leq \sum_{k \in TE} x_{ik} \geq 2$$

$$\sum_{k \in DST} x_{ik} = 1$$

$$\sum_{k=1}^{P} c_k x_{ik} \leq 50,000 \text{ (Budget Constraint)}$$

$$\sum_{k=1}^{P} x_{ik} = 9 \text{ (Lineup Constraint)}$$

$$x_{ik} \in \{0, 1\}, 1 \leq k \leq P$$

Other Constraints:

Defensive Stacking - This is a fairly intuitive result, but we add an additional constraint which forces the defense that we choose to not be opposing any of our offensive players. We denote the set of offensive players who are opposing a defense k by O_k.

$$6x_{ik} + \sum_{l \in O_k} x_{il} \leq 6, \forall k \in DST$$

QB-WR Stacking - Between any two positions on the field, the QB and his WR are the two most correlated in terms of fantasy points. In order to capture this correlation, we constrain that a QB/WR pair must exist in our lineup.

QB-oppWR Stacking - Although the basis of this is not rooted in correlations, we have noticed that winning lineups typically consist of a QB/WR/OppWR stack. This is due to us trying to capture as many fantasy points as we can from a high scoring game.

Exposure Constraint - To ensure we have diversity in our lineups, we add a positional exposure constraint such that no single player (given his position) can appear over q times.

No TE for Flex Constraint - A Flex player can be any RB/WR/TE, however TEs are generally the cheapest position, with the highest volatility and inflated projections. Having two bad TEs in a lineup ruins the chances of it scoring highly.

QB-TopWR Stacking - We stack a QB with a top receiver from the same team. WR must be 1 or 2 in terms of Targets Rank (see features).

Too Much Team Constraint - Cannot have 3 players from the same team.

Top Receivers Constraint - At least 3 receivers must be in the top 3 in Target Rank for their team. If Flex is WR, does not have to follow this rule.

Cheap Defenses Constraint - DST Salary $\geq$ 2700 and $\leq$ 3100

Current Best Strategy: Formulation 14 - Formulation 14 currently is our most

successful and has the following constraints. This consists of the following constraints:

- Feasibility Constraints
- Defensive Stacking
- QB-TopWR Stacking
- Exposure Constraint
- Cheap Defenses Constraint
- no TE for Flex Constraint

Chapter 5

Results

In this section, we present our findings on using asymmetric cost SVMs to predict which players exceed the efficiency threshold. To be succinct, we focus on the results that show a significant improvement over the baseline. WR is the only position for which asymmetric cost SVMs led to a significant increase in overall PnL.

5.1 Spiking WR Projections Using Asymmetric Cost SVM with Linear Kernel

5.1.1 Projections

Using weeks 4-16, we train an asymmetric cost SVM with a linear kernel using the heuristic described in Chapter 4. We find the optimal cost factor ratio to be 0.8, which yields a cross-validation error of 7.35% and a testing error of 20.59%. More importantly, the model labeled 16.47% of examples in the testing set as +1 and achieved a precision rate of 14.29% and a recall rate of 26.67%. The precision rate is interpreted as the fraction of players predicted to go off that did indeed go off, which can be thought of as an estimate of the conditional probability that a player goes off given that he is predicted to go off. The recall rate is interpreted as the fraction of players that go off that are correctly predicted to go off.

To avoid look-ahead bias, we train an asymmetric cost SVM classifier for each addi-

tional week starting from week 4 and ending in week 16. That is, we train the classifier using weeks 4-6 to predict week 7, using weeks 4-7 to predict week 8, and so on. The cost ratio, overall error, precision rate, and recall rate on the testing set for each week is listed in Figure 5.1.

Asymmetric Cost SVM with Linear Kernel				
Week	Cost Ratio	Overall Error	Precision	Recall
7	0.07	67.19%	17.31%	100%
8	0.1	15.71%	45.45%	50.00%
9	0.075	57.81%	12.5%	71.43%
10	0.065	80.30%	5.36%	100%
11	0.06	61.11%	4.44%	66.67%
12	0.01	37.50%	9.09%	22.2%
13	0.075	43.59%	8.33%	75.00%
14	0.08	22.08%	15.38%	25.00%
15	0.075	28.39%	5.00%	20.00%
16	0.08	12.66%	42.86%	75.00%

Figure 5.1: Performance in Live Weekly Testing (Train on Weeks i-j, Test on Week j+1)

There is clearly a tradeoff between overall error and precision as well as overall error and recall. Though the overall error is quite high in some weeks, we are willing to sacrifice accuracy for higher precision and recall. In Section 5.1.2, we will see that spiking the expert projections for players that the model predicts will go off significantly increases PnL.

5.1.2 Combinatorial Optimization

Using the greedy algorithm, we generate sets of 150 lineups using our new projections for weeks 7-16. For weeks 2-6, we use Formulation 14 (see Section 4.2) generate lineups with the original Daily Fantasy Nerd projections. To evaluate our results, we also generate

lineups for weeks 7-16 using the original Daily Fantasy Nerd projections. This is our baseline. Weekly PnLs are displayed in Figure 5.2 and Figure 5.3.

Week	PnL ($)
2	853
3	18,538
4	485
5	-179
6	-159
7	-222
8	220
9	-196
10	-150
11	170
12	-164
13	29
14	1,087
15	-341
16	382
Total	20,353

Figure 5.2: Baseline (Original Projections)

Week	PnL ($)
2	853
3	18,538
4	485
5	-179
6	-159
7	-59
8	179
9	-284
10	99
11	112
12	-30
13	27
14	1,000
15	-371
16	1,953
Total	22,164

Figure 5.3: Spiked Projections

Over the course of the season, using the spiked projections makes $1,811 (8.9%) more than the baseline. If we remove week 3, which likely was an outlier due to exceedingly good projections, the PnLs are $1,815 using the original projections and $3,626 using the spiked projections—an improvement of 99.8%. Considering only the weeks where we spiked projections (weeks 7-16), the PnLs are $815 using the original projections and $2,626 using the spiked projections, which is a 322% increase.

We also evaluate the performance of our strategy by comparing the distribution of fantasy points earned by generated lineups in each week. We count the number of lineups

in the following fantasy point ranges: 160-169.9, 170-179.9, 180-180.9, 190-199.9, and 200+. In all weeks of the 2016-2017 season, scoring above 160 fantasy points was enough to cash. Plots of the weekly counts for the baseline and spiked projection strategies are displayed in Figure 5.4 and Figure 5.5, respectively.

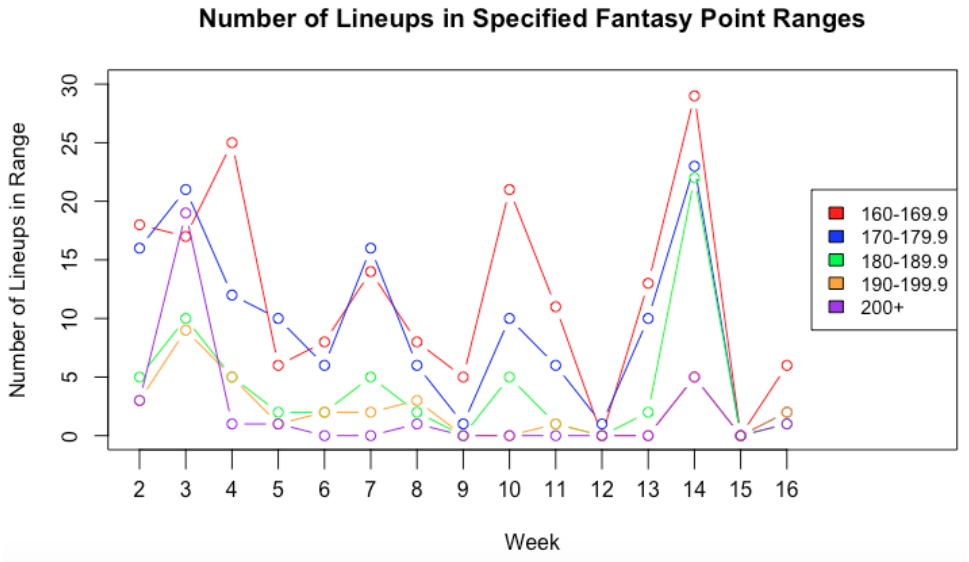

Figure 5.4: Baseline (Original Projections)

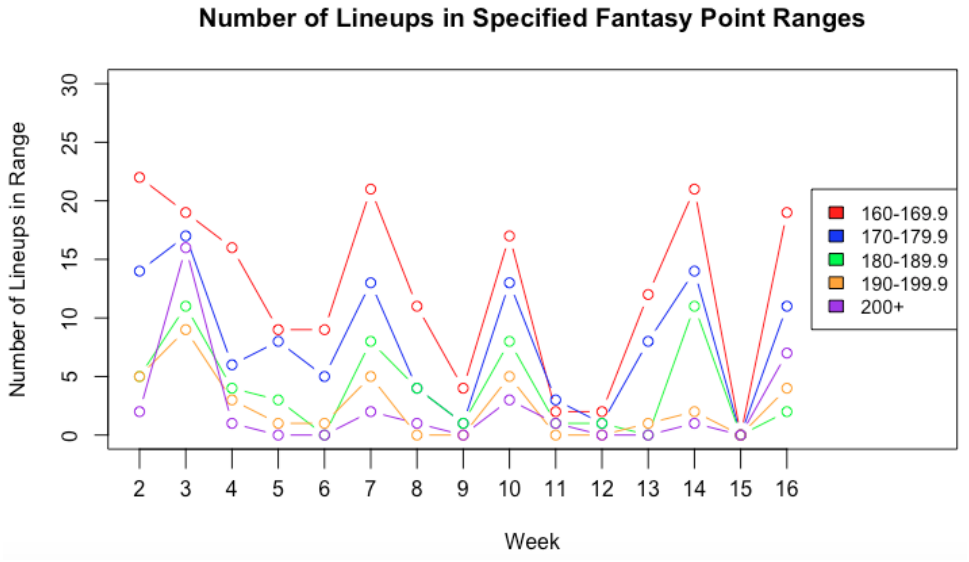

Figure 5.5: Spiked Projections

While the number of lineups in the 160-169.9 fantasy points range is oftentimes higher for the baseline strategy, notice that there are multiple weeks (7, 10, and 16) where the number of lineups in the 190-199.9 and 200+ fantasy points ranges is higher for the spiked

projection strategy. Although the number of fantasy points required to earn a big payday fluctuates from week to week, scoring multiple lineups in the 190-199.9 and 200+ range is generally enough to make a decent profit. This is consistent with the results in weeks 7, 10, and 16, where spiking projections increases the baseline PnL by \$163, \$249, and \$1,571, respectively.

We further evaluate the baseline and spiked projections strategies by comparing the distributions of lineup fantasy points aggregated over all weeks. In order to do so, we estimate the probability distribution function of lineup fantasy points for the two strategies via kernel density estimation. We remove lineups that score less than 160 fantasy points to hone in on cashing lineups. In addition, we conduct a bootstrap hypothesis test of equality and compute the corresponding confidence band ($\alpha = 0.05$). The overlaid kernel density plots are displayed in Figure 5.6.

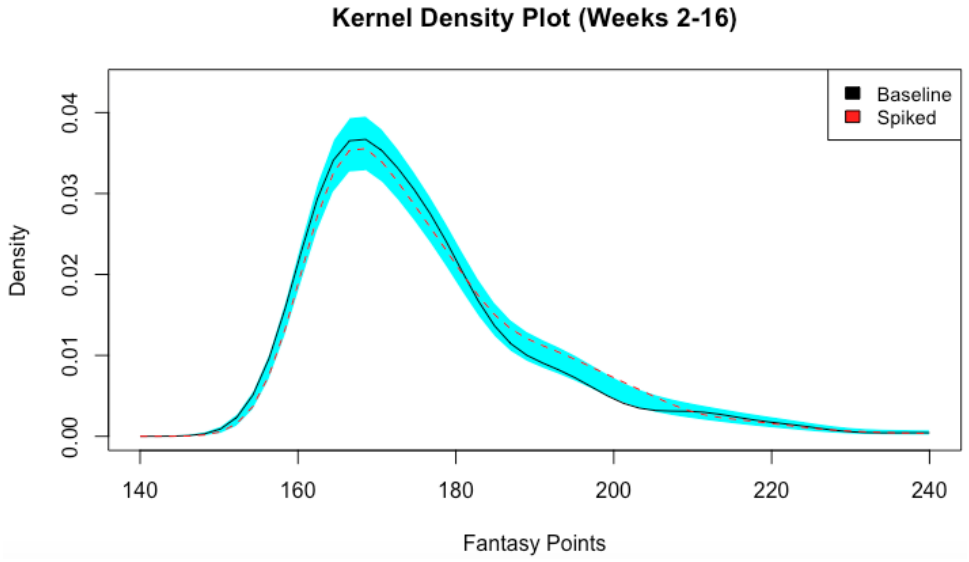

Figure 5.6: Kernel Density Plot of Fantasy Points for Lineup Scoring Over 160 in Weeks 2-16. Baseline (Black), Spiked Projections (Dotted Red), and 95% Confidence Band (Cyan).

Notice that the red dotted line (spiked projections strategy) is visibly higher than the black line (baseline strategy) in approximately the 180-210 fantasy points range. For the 200-204 fantasy points range (which earns a decent payout in most weeks), the difference between the two strategies is statistically significant. This suggests that, compared to the baseline strategy, our strategy of spiking projections based on the predictions of the

asymmetric cost SVM is more likely to produce high scoring lineups with big payouts.

Chapter 6

Conclusion

6.1 Discussion

As mentioned at the end of Chapter 5, we managed to increase profits by 322% in the weeks where we spiked projections (7-16) based on the predictions of the asymmetric cost SVM. In some weeks (8 and 16), we manage to increase precision from 0% to over 40% using the asymmetric cost SVM. Simply using the projections provided by "expert" sites rarely yields lineups that have a full set of 9 players that score highly; we are able to achieve so much success using this model because it increases the probability of generating a lineup where every player scores highly. Due to the top-heavy payout structure of the "NFL $1M Play-Action" GPP contest, having one or two high scoring lineups can increase PnL dramatically, so trading accuracy for precision is favorable.

Though we managed to dramatically increase the total profit of weeks 7-16, it is important to notice that the majority of profits are the result of an exceptional week 16. In week 16, there are several high scoring lineups, leading to a profit of $1,952 (versus baseline profit of $382). If we compare the two strategies for weeks 7-15, the total profit of the spiked projections strategy is 55% higher than the total profit of the baseline strategy, rather than 322%. The fact that the majority of profits are the result of a single week is not surprising—our strategy increases the probability of generating high scoring lineups, and the payout structure of GPPs are very top heavy.

It is somewhat surprising that our strategy makes more money than the baseline strategy even after removing week 16. We can think of two explanations for the robustness of our strategy. One explanation is that our strategy increases the probability of generating high scoring lineups sufficiently high such that profits are increased significantly in multiple weeks. Thus, removing one good week does not make the baseline strategy better. Another explanation is based on the way we solve the combinatorial optimization problem—since we solve a greedy integer program that maximizes the projected number of fantasy points, most of the players with low projections will not be selected, even if their projections are increased by a factor of 1.5. As a result, the difference between the set of lineups generated using our strategy and the baseline strategy is usually just a few players and the percentage of lineups in which they show up. Since the players that the asymmetric cost SVM mislabels as +1 are oftentimes selected in the baseline strategy, correctly labeling even one player as +1 and getting exposure to that player in lineups generally leads to an improvement on the baseline.

6.2 Future Work

In the future, we will look into more ways we can utilize machine learning algorithms to predict which players will go off. For example, we could apply the same methodology we employed in this paper to predict which set of relatively expensive players will score exceedingly high. In order to do so, we would define an efficiency threshold for expensive players, much like we did for cheap players. Obviously, this threshold would be much higher—expensive players occasionally score 50+ fantasy points in a single week (e.g. WR Julio Jones in week 4 and Le'Veon Bell in week 14). We would once again have a small number of positive labels in the response vector, so asymmetric cost SVM may be a good approach for this problem as well.

We can also look into competing in Double-Up contests. Many DFS professionals play both GPPs and Double-Ups because the former provides an opportunity to win a large sum of money while the latter provides a more steady stream of cash from week

to week. The approach we would take for Double-Up contests differs from our approach for GPPs. For GPPs, we are trying to maximize the number of lineups that score big because the payout structure is top heavy. In contrast, for Double-Ups, we are trying to maximize the number of lineups that exceed the cutoff score (50th percentile among all entries for the contest) because the payout structure is binary. One possible approach for winning Double-Ups is to predict the set of players that will exceed some "inefficiency threshold"—any player below this threshold scores too few fantasy points, which we want to avoid as much as possible in Double-Ups. Along these lines of finding steady players that will not perform poorly, another approach is to compute the variance for each player and construct lineups that maximize the mean and minimize the variance.

Finally, we may view the problem of choosing a set of optimal players from a game theory perspective. If every contestant submitted the same lineups, they would all receive or lose the same amount of money. In GPPs, this amount is always negative. In order to beat competitors, we must submit lineups that are strictly better than those of competitors—thus, each player's "ownership rate" is crucial to doing well in contests. For example, consider two players, A and B, where the former appears in 5% of lineups while the latter appears in 25% of lineups. On the one hand, if player A scores 1 fantasy point higher than player B, a lineup with player A rather than player B (all else equal) will place higher than 25% of lineups. On the other hand, if player B scores 1 fantasy point higher than player A, a lineup with player B rather than player A (all else equal) will place higher than only 5% of lineups. Thus, if the projections for players A and B are equal and we have no additional information about which player will outperform the other, it is optimal to select player A for our lineups. One approach to integrating this game theory perspective is to train a model that predicts the ownership rate of each player given our feature set; if the projections of two players are equal, we slightly increase (e.g. +0.1) the projection of the player with the lower ownership rate. Another approach is to increase or decrease all projections by some factor based on ownership rate.

Bibliography

[1] Lutz, R. (2015). Fantasy Football Prediction. Retrieved January 25, 2017, from https://arxiv.org/pdf/1505.06918v1.pdf.

[2] Parikh, N. (2015). Interactive Tools for Fantasy Football Analytics and Predictions using Machine Learning. Retrieved January 25, 2017, from https://dspace.mit.edu/handle/1721.1/100687#files-area.

[3] Dunnington, N. (2015). Fantasy Football Projection Analysis. Retrieved January 25, 2017, from http://economics.uoregon.edu/wp-content/uploads/sites/4/2015/03/Dunnington_Thesis_2015.pdf.

[4] Hunter, D., Vielma, J., Zaman, T. (2016). Picking Winners Using Integer Programming. Retrieved January 25, 2017, from http://www.mit.edu/ jvielma/publications/Picking-Winners.pdf.

[5] Morik, K. (1999). Combining Statistical Learning with a Knowledge-Based Approach—A Case Study in Intensive Care Monitoring. Retrieved January 25, 2017, from http://www.cs.cornell.edu/People/tj/publications/morik_etal_99a.pdf.

[6] Joachims, T. (2008). Making Large-Scale SVM Learning Practical. Retrieved January 25, 2017, from http://svmlight.joachims.org/.