

# Daily Fantasy Football

By

Jason R. Becker & Jack St. Clair

A report submitted as an independent study project for the  
Master of Financial Engineering Program at the  
University of California, Berkeley

Spring 2019

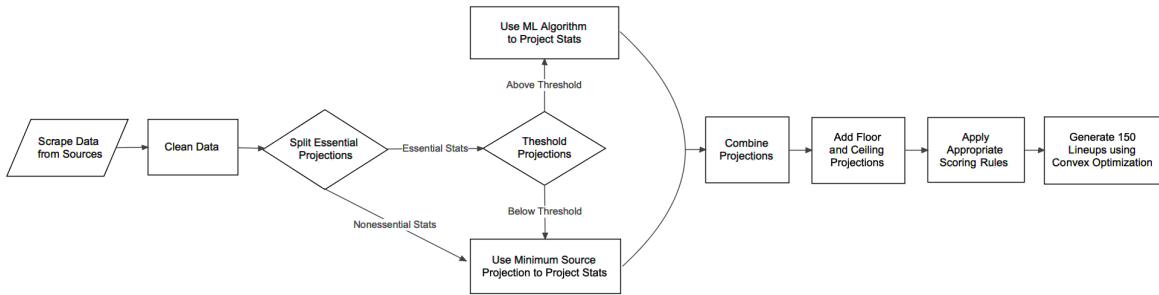


---

# 1 Introduction

Popularity of daily fantasy sports (DFS) betting markets surged in the early 2010s reaching mainstream popularity by 2015. As legal sports gambling becomes legal in many states across the US, the amount of money bet in DFS and season-long fantasy leagues should only increase as a result. More than \$5 Billion was bet on sports in general throughout 2018 in the state of Nevada alone, breaking the previous record of nearly \$5 Billion from 2017 ([1], [2]). And, the fantasy sports market is now reported to be a \$7 Billion Industry [3]. The focus of this paper is on DFS in the National Football League (NFL), specifically detailing a methodology for collecting data, projecting points, and generating optimal lineups for use in competition at the major DFS sites, DraftKings and FanDuel. Both of these sites offer weekly contests during the NFL season in which the winner receives \$1 Million, costing only \$20 to enter per lineup. The lineups generated will be tested using the results from these \$1 Million prize contests.

As a parallel to finance, DFS lineup generation closely mirrors portfolio optimization, as lineups (portfolios) are optimized to maximize points (returns) subject to a number of lineup (portfolio) constraints such as limiting the number of players (assets) in total and within a particular position (asset class). As in financial practice, we will utilize convex optimization to generate our lineups. The full end-to-end pipeline used to generate lineups for daily fantasy sites is provided in Figure 1. This paper will discuss the sections of this pipeline in chronological order: from scraping and cleaning the data, to generating stat and scoring projections, through generating final lineups for each week.



**Figure 1:** Full data pipeline for generating optimal lineups.

## 2 Data Summary

### 2.1 Data Collection

**Table 1:** Data collected from each source.

Source	Full Season	Weekly
CBS	QB, RB, WR, TE, DST, K	QB, RB, WR, TE, DST, K
ESPN	QB, RB, WR, TE, DST, K	QB, RB, WR, TE, DST, K
FantasyPros	QB, RB, WR, TE, DST, K	QB, RB, WR, TE, DST, K
FFToday	QB, RB, WR, TE, DST, K	QB, RB, WR, TE, K
NFL.com	QB, RB, WR, TE, DST, K	QB, RB, WR, TE, DST, K
RTSports	QB, RB, WR, TE, DST, K	
Yahoo		QB, RB, WR, TE, DST, K
DraftKings		QB, RB, WR, TE, DST
FanDuel		QB, RB, WR, TE, DST

For the scope of this report, full season analysis will be neglected. However, the data has been collected, allowing for potential future work on full season projections which may prove useful for season long fantasy leagues. As daily fantasy websites such as DraftKings and FanDuel do not include kickers in possible lineups, kickers will also be neglected for the remainder of this report.

For each position, a variety of stats are projected from the various sources. Due to lack of consistency between which stats are projected and which are not among the sources, stats were broken down into two categories. The first category we denote *essential stats*, stats which are projected by most of the sources and are generally non-zero in value. The second category we denote *nonessential stats*, stats which may not be projected by more than a couple sources and are generally 0 or near 0 in value. Table 2 lists the essential and nonessential stats for each position.

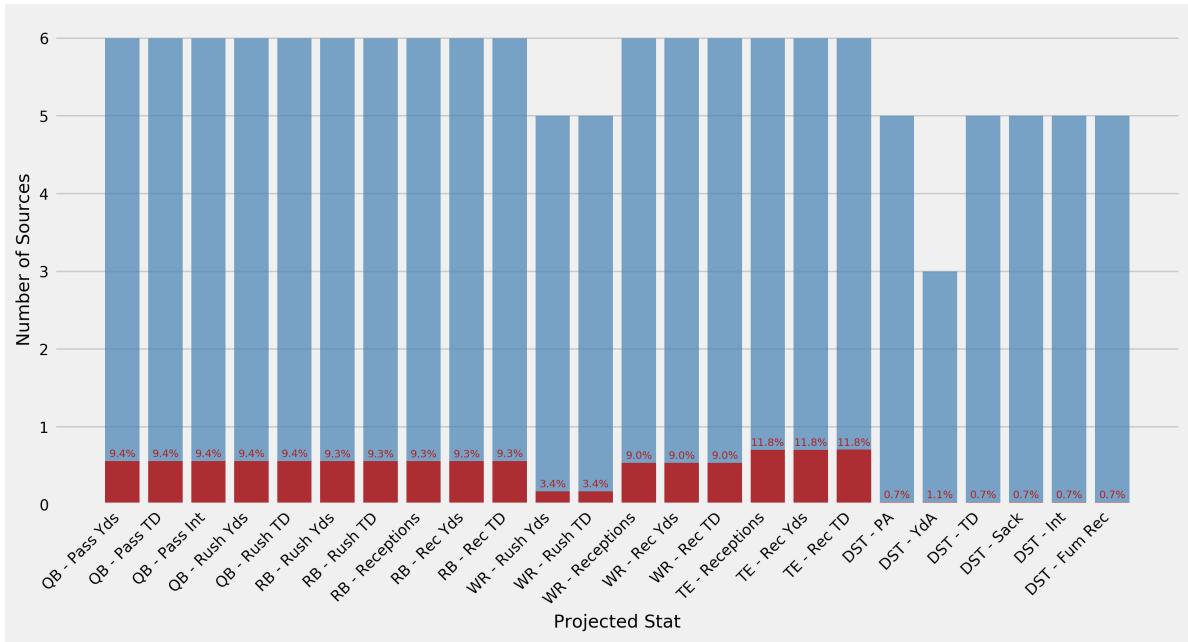
**Table 2:** Essential and nonessential stats for each position.

Position	Essential Stats	Nonessential Stats
QB	Pass Yds, Pass TD, Pass Int, Rush Yds, Rush TD,	Receptions, Rec Yds, Rec TD, 2PT
RB	Rush Yds, Rush TD, Receptions, Rec Yds, Rec TD	Pass Yds, Pass TD, Pass Int, 2PT
WR	Rush Yds, Rush TD, Receptions, Rec Yds, Rec TD	Pass Yds, Pass TD, Pass Int, 2PT
TE	Receptions, Rec Yds, Rec TD	Pass Yds, Pass TD, Pass Int, Rush Yds, Rush TD, 2PT
DST	PA, YdA, TD, Sack, Int, Fum Rec	Saf, Blk

### 2.2 Missing Data

Among data collection sources, the number of players with projections was not consistent, and in many cases projections are only provided from 4 or 5 out of 6 sources for a particular

player. The number of sources which provided projections for each essential stat, and the percentage of missing data are provided in Figure 2. Nonessential statistics are provided in the appendix. These instances were manually observed and found not to be restricted to backups, but often occur in prominent starting players (e.g., Drew Brees, Cam Newton). Since prominent players were affected, the players with missing projections could not be dropped, necessitating the imputation of missing values.

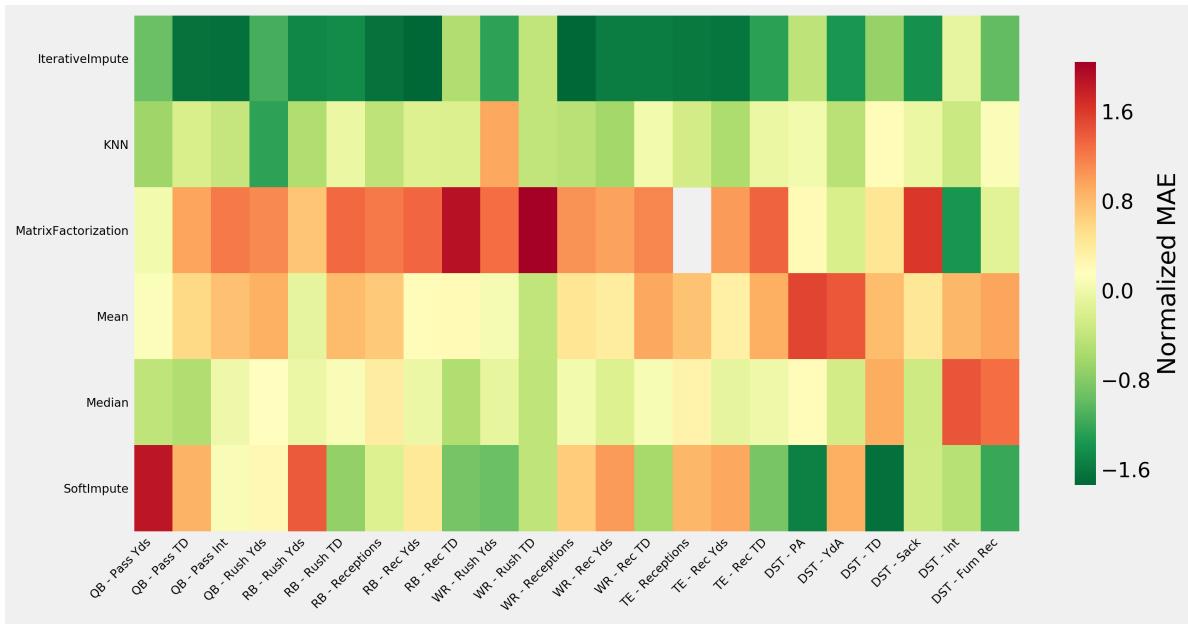


**Figure 2:** Number of sources collected for each essential stat (blue). Red indicates percentage of missing data for each respective stat.

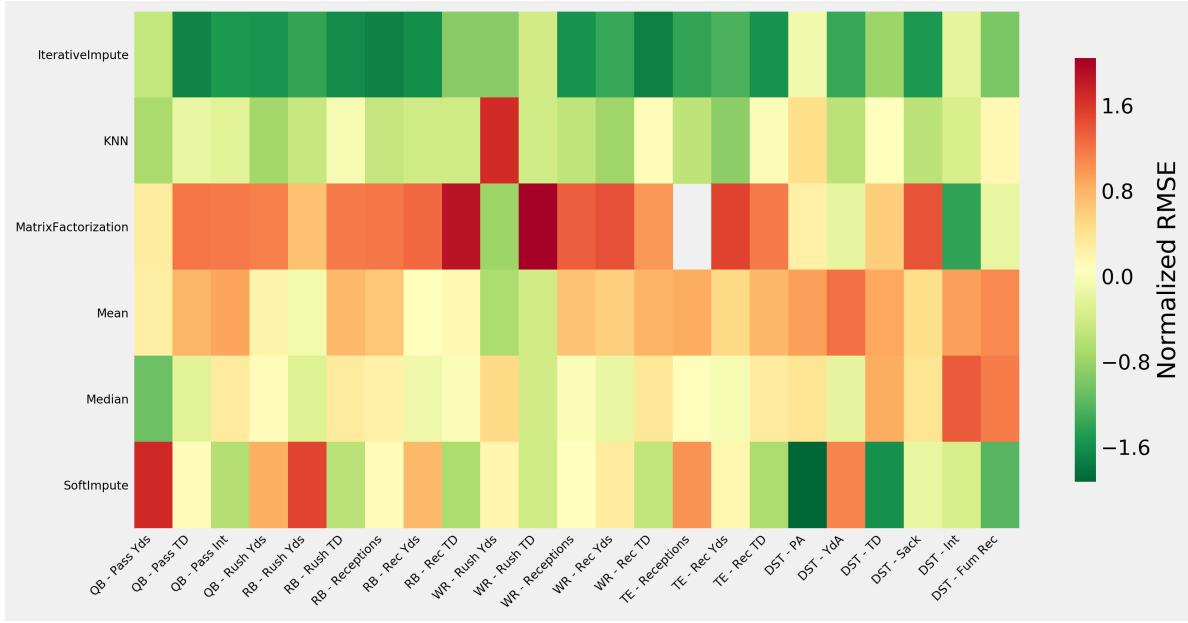
Several imputing methods were tested to find the optimal method for imputing missing projections. Imputing algorithms from the `fancyimpute` Python package were utilized to perform the imputing [4]. The simplest of these techniques is simply filling the projections from missing sources with the mean or median of the collected projections. Several more advanced methods were also studied. First, an iterative imputing method was used, where each source with missing values was first modeled as a function of all other sources using rows with full data, and these respective models were subsequently used to impute missing values for each source. Second, a K-nearest neighbors (KNN) approach was used, where each source is given a weight based on rows with all data observed. A matrix factorization approach was also studied which uses gradient descent to directly solve a factorization of the incomplete matrix into low-rank  $U$  and  $V$  matrices with L1 and L2 regularization respectively. Finally a soft impute method was studied which completes the matrix via soft thresholding singular value decompositions (SVD). Strictly using SVD was not possible due to the low number of sources (6) compared to the number of players which ranged from 32 for DST to upwards of 200 for WR. More information regarding each method and literature is available at [4].

In order to test which imputing method was best suited for the stat projection dataset, a raw matrix of all projected players for every week of the season was compiled for each

stat. Players with 50% or more sources missing were dropped, as these players were always either backups or players not expected to start, resulting in an incomplete matrix. Next, the percentage of missing data in the incomplete matrix for each respective stat was recorded, shown in Figure 2. All rows (players) with any missing values were subsequently removed, leaving a fully completed matrix for each stat. Elements in this completed matrix were then removed at random until the matrix had an equivalent missing data percentage as incomplete matrix. Each imputation method described above was then used to impute these missing values, with the mean average error (MAE) and root mean squared error (RMSE) recorded for the difference between the imputed matrix of each method and the complete matrix of true projection values. This process was repeated for 50 simulations in order to reduce potential variation from the random selection of missing elements. The resulting MAE and RMSE values for each imputing method of the essential stats is provided below.



**Figure 3:** Normalized MAE of imputing methods for each essential stat.

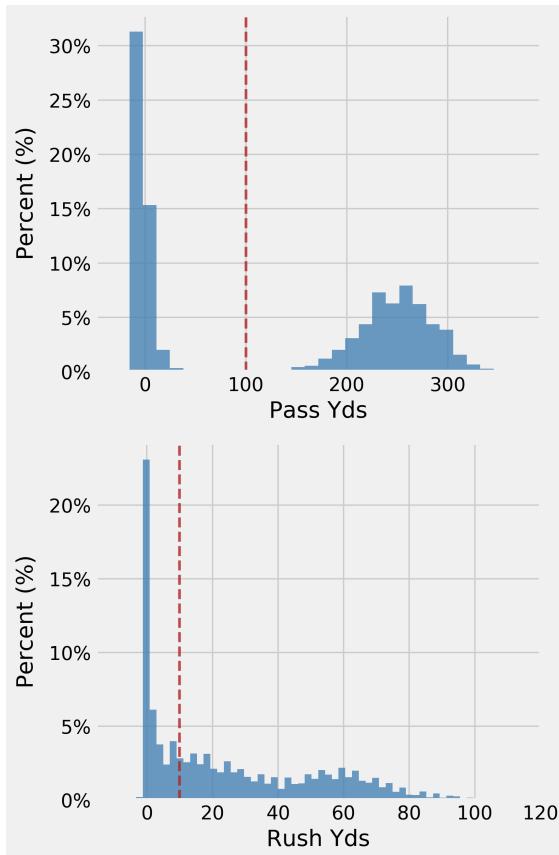


**Figure 4:** Normalized RMSE of imputing methods for each essential stat.

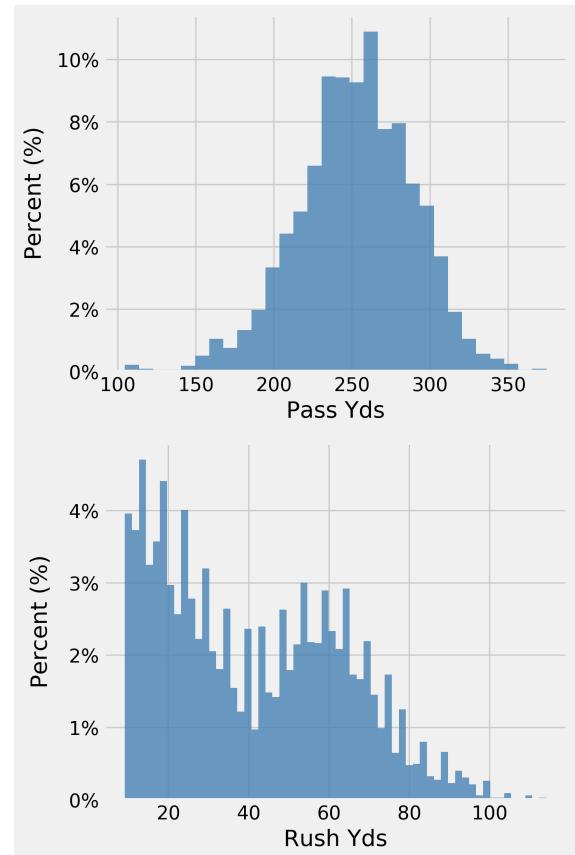
Figures 3 and 4 clearly indicate that the iterative imputing method provides the best estimates for missing values. A comparison of imputing techniques for nonessential stats is provided in the appendix, where iterative imputing was again found to be optimal. Therefore iterative imputing was used to impute missing values for both essential and nonessential stats for the remainder of this study.

## 2.3 Data Exploration

The raw values for realized stats were investigated prior to building predictive models. For each stat, two nearly independent distributions were observed, one near zero and the other significantly different than zero. These two distributions were identified as *starters* and *backups* (i.e., players who were expected to play the entire game and players who come off the bench and play a minimal role respectively). Due to the difficulties that would arise from machine learning models attempting to predict multiple distributions, a threshold was set for each stat to separate values from starters and backups. Separate models were used to predict stats above and below these thresholds, both of which will be discussed further in the methodology section. Thresholds were selected manually through observation of histograms of each stat. Figure 5a displays an example threshold for an easily separable stat, Pass Yds, where the two distributions are clearly distinguishable. Figure 5b displays an example threshold for a more difficult stat to separate, Rush Yds, which appears to have three distributions, one near zero, one linear decreasing, and a third bell shape distribution. For stats such as Rush Yds, the separation boundary was chosen to separate near-zero and non-zero distributions to avoid having numerous categorical distributions within each stat. The selected thresholds and resulting histograms after thresholding for all essential stats are provided in the appendix.



(a) Raw histogram with threshold (red).



(b) Histogram above threshold.

**Figure 5:** Example raw and thresholded histograms for QB passing yards and RB rushing yards.

## 3 Methodology

### 3.1 Projections

In order to project scores for each player, the individual stats of that player were projected, and subsequently combined with scoring rules for either FanDuel or DraftKings to generate a final scoring projection for each week. Models used for individual stat projections were trained on a *training* period (Weeks 1-11), validated on a *cross-validation* period (Weeks 12-14), and finally tested on an unseen *testing* period (Weeks 15-17). This train/cross-validation/test procedure is the standard in machine learning literature as it ensures models are not overfit to the training data, establishes an unseen period to identify the best model in the cross-validation period, and can be tested on completely unseen data in the test period, providing a fair estimate for how well the model will do given future data.

For all stats, the projections from individual sources were used as feature variables for the final projection. More concretely, for a given stat such as Pass Yds, the feature variable vector  $X = [\text{CBS}, \text{ESPN}, \text{FantasyPros}, \text{FFToday}, \text{NFL}, \text{RTSports}, \text{Yahoo}]$  where each source name represents the Pass Yds projection from that respective source. Given the feature variable vector, the projection  $y$  can be found:

$$y = f(X) \quad (1)$$

where  $f(\cdot)$  can be any function. The studied functions used to form projections are given in Table 3.

**Table 3:** Functions and algorithms used to project stats from feature variables.

Abbreviation	Name	Source
Min	Minimum	-
Max	Maximum	-
Mean	Mean	-
Median	Median	-
LR	Linear Regression (Elastic Net)	[5, 6]
XGB	eXtreme Gradient Boosted Trees	[7, 8]
OLS	Ordinary Least Squares	[9]
RF	Random Forest Regressor	[10, 11]
SVR	Support Vector Regressor	[12, 13]

For nonessential stats, only simple methods were compared, as most stats did not have enough sources providing projections to incorporate machine learning models. The MAE results for these simple methods are provided below. As the minimum projection provided the lowest MAE for each stat with multiple projections, it was selected as the projection function for all nonessential stats.

	Min	Max	Mean	Median
QB - Receptions				
QB - Rec Yds				
QB - Rec TD				
QB - 2PT				
RB - Pass Yds				
RB - Pass TD				
RB - Pass Int	0.001	0.009	0.004	0.001
RB - 2PT				
WR - Pass Yds				
WR - Pass TD				
WR - Pass Int				
WR - Rush TD	0.004	0.009	0.005	0.005
WR - 2PT				
TE - Pass Yds				
TE - Pass TD				
TE - Pass Int				
TE - Rush Yds	0.030	0.105	0.053	0.040
TE - Rush TD	0.001	0.001	0.001	0.001
TE - 2PT				
DST - Saf	0.027	0.079	0.046	0.039
DST - Blk	0.078	0.126	0.102	0.102

**Figure 6:** Training set MAE of simple projection methods for nonessential stats.

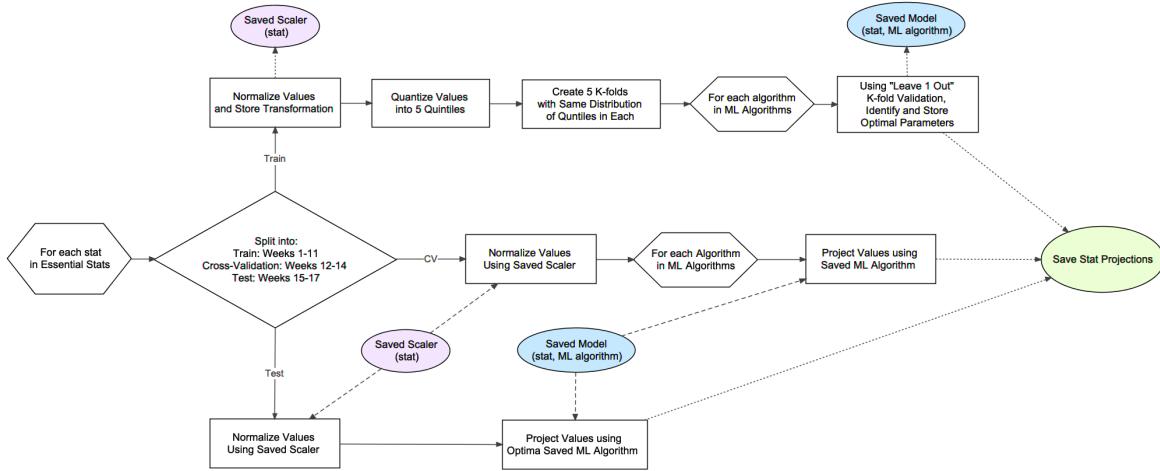
Similarly, essential stats below the threshold values were projected using only simple methods. Since the values were all near-zero, the added complexity of using more advanced methods would not result in much benefit to the final scoring projection for each player. As with nonessential stats, the minimum projection for stats below the specified threshold provided the lowest MAE, so minimum was the function selected for projection of stats when the mean of source projections was below the respective specified stat threshold.

	Min	Max	Mean	Median
QB - Pass Yds	5.2	19.8	10.2	9.0
QB - Pass TD	0.2	0.3	0.2	0.2
QB - Pass Int	0.2	0.4	0.3	0.3
QB - Rush Yds	1.3	3.1	2.1	2.0
QB - Rush TD	0.0	0.1	0.0	0.0
RB - Rush Yds	4.2	11.1	7.1	6.9
RB - Rush TD	0.1	0.3	0.1	0.1
RB - Receptions	0.2	1.0	0.6	0.6
RB - Rec Yds	2.8	9.2	5.7	5.7
RB - Rec TD	0.0	0.1	0.0	0.0
WR - Rush Yds	0.3	1.1	0.6	0.6
WR - Receptions	0.6	1.4	1.0	0.9
WR - Rec Yds	6.9	18.2	12.0	11.6
WR - Rec TD	0.1	0.3	0.2	0.2
TE - Receptions	0.5	1.0	0.7	0.7
TE - Rec Yds	4.1	11.5	7.6	7.6
TE - Rec TD	0.0	0.2	0.1	0.1
DST - PA	1.9	2.2	2.0	2.0
DST - YdA	41.2	50.2	45.5	45.0
DST - TD	0.1	0.2	0.2	0.2
DST - Sack	1.0	1.5	1.2	1.2
DST - Int	0.6	0.9	0.7	0.7
DST - Fum Rec	0.4	0.8	0.6	0.5

**Figure 7:** Training set MAE of simple projection methods for essential stats below respective stat thresholds.

Finally, essential stats above the specified thresholds were projected using the full set of studied algorithms. First, OLS was fit to the training set for each stat, which provides a simple linear approximation for the influence of each feature variable on the final projection  $y$ . OLS is often used as a base machine learning model since it is the simplest model with small chance of overfitting. OLS results are reported in the appendix, where the significance of each individual source for every stat is additionally provided. Note that the significance indicated is only for the *linear* relation between each respective source and the true realized stat values, and sources that are deemed insignificant by OLS may still be significantly related to the true value by another degree (e.g., ESPN’s projection for Pass Yds may be insignificant linearly but the square of ESPN’s projection may be informative).

In order to find these more complex relationships between source projections and realized stat values, the machine learning algorithms detailed in Table 3 were implemented. More information about the individual machine learning algorithms can be found in the cited references. Each stat was treated as a separate dataset, with the methodology outlined below applied to each individually as shown in Figure 8.



**Figure 8:** Pipeline of machine learning generated projections. Color is used to show when models are saved and loaded.

Each machine learning model was trained using the training set of data (Weeks 1-11). Since several of the models tend to provide better estimates on similarly scaled data, the data was normalized using the training set mean and standard deviation for each stat:

$$X_{scaled} = \frac{X - \mu(X)}{\sigma(X)} \quad (2)$$

where  $\mu$  and  $\sigma$  denote the mean and standard deviation of  $X$  projections for each source respectively. This scaling results in each  $X_{scaled}$  having mean 0 and standard deviation of 1.

Once the values for each stat were scaled, they were quantized into quintiles (i.e., given a categorical value 1-5 depending on if they were in the 0-20, 20-40, ..., 80-100 percentile of projections). Next, the full training data was randomly split into 5 subsets, each of which had an identical number of values from each quintile, known as a stratified shuffle split. Stratifying the split using quintiles ensures that each subset had a similar distribution of high, medium, and low values, balancing each subset. Next, a randomized cross-validation grid search was used on the first 4 subsets to find an optimal model, and tested on the fifth subset to observe how the model performed on unseen data. This was repeated for each group of 4 subsets, known as K-fold cross-validation, to help the models generalize to unseen data. Optimal model hyperparameters from the randomized grid search were chosen from the hyperparameters which performed best on average over each of the 5 K-folds, where MAE was used as loss function. Once model hyperparameters were found for each model, they were saved to files such that they could be loaded and applied to other datasets.

After all model hyperparameters were found for each stat, the cross-validation set (Weeks 12-14) was loaded. The saved scaling function mean and standard deviation from the training sets were then applied to scale the cross-validation data in the same manner as the training data. Following, all machine learning models were loaded and applied to the scaled data, generating projections on data previously unseen by the machine learning models which had

no impact on hyperparameter selection. The resulting MAE for each stat and projection function is provided in Figure 9.

	Min	Max	Mean	Median	LR	SVR	OLS	XGB	RF
QB - Pass Yds	66.33	63.70	62.78	62.90	61.94	62.30	61.40	59.04	55.86
QB - Pass TD	0.91	0.77	0.83	0.83	0.81	0.80	0.82	0.79	0.80
QB - Pass Int	1.03	0.61	0.73	0.71	0.71	0.77	0.70	0.71	0.74
QB - Rush Yds	11.38	10.51	10.17	10.28	9.91	8.67	10.01	9.13	9.25
QB - Rush TD	1.10	0.89	0.99	0.99	0.96	1.01	0.96	0.94	0.95
RB - Rush Yds	26.35	25.34	24.48	24.51	24.20	24.79	24.32	23.86	23.95
RB - Rush TD	1.05	0.60	0.85	0.87	0.84	0.95	0.84	0.79	0.83
RB - Receptions	1.64	1.45	1.40	1.39	1.38	1.48	1.38	1.37	1.35
RB - Rec Yds	18.27	16.07	16.02	16.04	16.05	16.59	16.10	16.15	14.94
RB - Rec TD	1.09	0.89	0.99	0.99	0.96	0.97	0.96	0.94	0.95
WR - Rush Yds	13.80	11.20	12.48	12.64	13.00	12.84	12.85	12.95	13.14
WR - Receptions	1.84	1.66	1.61	1.60	1.59	1.63	1.59	1.59	1.56
WR - Rec Yds	25.12	25.08	23.77	23.79	23.66	23.90	23.66	23.33	22.63
WR - Rec TD	0.99	0.56	0.81	0.83	0.81	0.92	0.81	0.75	0.79
TE - Receptions	1.77	1.34	1.47	1.47	1.49	1.53	1.50	1.51	1.45
TE - Rec Yds	18.98	17.61	17.44	17.48	17.46	17.34	17.47	16.95	16.41
TE - Rec TD	0.99	0.64	0.84	0.86	0.85	0.91	0.85	0.79	0.82
DST - PA	7.65	7.42	7.30	7.24	7.19	6.60	7.17	6.72	6.73
DST - Yda	68.50	63.80	63.55	63.86	63.01	63.15	63.05	61.33	61.70
DST - TD	1.01	0.85	0.93	0.92	0.94	1.00	0.94	0.96	0.92
DST - Sack	1.36	1.27	1.29	1.30	1.26	1.18	1.26	1.25	1.23
DST - Int	0.83	0.57	0.66	0.66	0.67	0.54	0.67	0.67	0.66
DST - Fum Rec	0.81	0.45	0.63	0.65	0.72	0.81	0.72	0.72	0.64

	Min	Max	Mean	Median	LR	SVR	OLS	XGB	RF
QB - Pass Yds	64.76	71.64	66.51	66.67	66.80	66.13	67.06	66.35	66.01
QB - Pass TD	0.81	0.75	0.75	0.74	0.75	0.76	0.75	0.77	0.74
QB - Pass Int	1.04	0.73	0.81	0.80	0.80	0.93	0.82	0.79	0.86
QB - Rush Yds	15.67	14.64	14.56	14.74	14.72	14.94	14.76	14.97	13.96
QB - Rush TD	0.98	0.66	0.81	0.80	0.81	0.91	0.81	0.80	0.83
RB - Rush Yds	26.04	24.54	24.17	24.11	23.95	24.98	23.90	24.97	24.75
RB - Rush TD	0.97	0.49	0.78	0.80	0.79	0.91	0.79	0.78	0.79
RB - Receptions	1.68	1.60	1.52	1.53	1.50	1.59	1.50	1.50	1.51
RB - Rec Yds	16.22	16.05	15.46	15.59	15.51	15.65	15.56	15.68	15.48
RB - Rec TD	0.98	0.74	0.86	0.87	0.84	0.89	0.84	0.85	0.87
WR - Rush Yds	16.08	13.75	14.40	14.53	15.24	15.45	15.22	15.53	15.50
WR - Receptions	1.86	1.74	1.68	1.69	1.67	1.70	1.67	1.68	1.64
WR - Rec Yds	25.29	26.35	24.70	24.76	24.49	24.35	24.51	24.30	23.98
WR - Rec TD	0.97	0.58	0.79	0.80	0.77	0.89	0.77	0.76	0.77
TE - Receptions	1.88	1.48	1.62	1.62	1.65	1.78	1.67	1.67	1.62
TE - Rec Yds	20.05	19.26	18.49	18.55	18.52	19.17	18.77	19.10	18.69
TE - Rec TD	0.96	0.53	0.80	0.83	0.84	0.91	0.84	0.82	0.84
DST - PA	6.21	6.42	6.09	6.14	6.17	6.08	6.19	6.09	5.99
DST - Yda	60.00	63.58	61.16	61.86	60.72	60.32	60.90	61.49	60.50
DST - TD	1.00	0.84	0.92	0.92	0.93	1.00	0.93	0.96	0.92
DST - Sack	1.17	1.08	1.06	1.06	1.04	1.04	1.04	1.05	1.03
DST - Int	0.92	0.74	0.78	0.77	0.78	0.74	0.81	0.80	0.81
DST - Fum Rec	0.77	0.46	0.62	0.62	0.68	0.77	0.68	0.70	0.63

(a) Training MAE.

(b) Cross-Validation MAE.

**Figure 9:** Mean Absolute Error (MAE) values for all essential stats.

Models with low MAE values on the training set but moderate to high MAE values on the cross-validation set indicate that the models overfit to the training set, and do not generalize well to unseen data. The optimal models for projecting each stat were manually chosen based on the MAE results from both the training and cross-validation sets. For stats such as DST - Fum Rec where both the cross-validation and training sets had the same optimal model (Max), that model was selected. For stats such as QB - Rush Yds, were the best model on the cross-validation set (RF) differed from the training set (SVR), the model performing best on the cross validation set was generally chosen. Some exceptions to these rules were made. For example, with QB Pass TD, although the RF slightly outperformed the Max on the cross-validation set, it significantly underperformed compared to Max on the training set. This indicates that the model may have gotten “lucky” on the cross-validation set and similar performance greater than the training set cannot be expected on new data. Additionally, when two models perform nearly equally well, a simpler model (Max) is preferable over a more complex model (RF). Hence, for QB - Pass TD, Max was chosen as the projection function.

Once the projection function was chosen for each stat, the test set data was loaded. Similar to the cross-validation set, the test set was scaled using the saved scaling data from the training set, and then the optimal projection function was applied to each stat, forming final projections for each stat and data that was completely unseen to models during hyperparameter tuning, and during the model selection process. As such, the test set provides a reliable estimate for the MAE expected for each stat on new data to the pipeline. The

selected projection function for each stat, as well as its respective MAE value on the training, cross-validation, and test sets are provided in Table 4.

**Table 4:** MAE values for training, cross-validation, and test sets with selected optimal projection function for each stat.

Position	Stat	Projection Function	Train	CV	Test
QB	Pass Yds	RF	55.86	66.01	61.36
	Pass TD	Max	0.77	0.75	0.79
	Pass Int	Max	0.61	0.73	0.48
	Rush Yds	RF	9.25	13.96	14.10
	Rush TD	RF	0.95	0.83	1.12
RB	Rush Yds	OLS	24.32	23.90	23.51
	Rush TD	Max	0.60	0.49	0.50
	Receptions	XGB	1.37	1.50	1.27
	Rec Yds	RF	14.94	15.48	14.08
	Rec TD	XGB	0.94	0.85	0.89
WR	Rush Yds	Max	11.20	13.75	10.00
	Receptions	RF	1.56	1.64	1.62
	Rec Yds	RF	22.63	23.98	22.54
	Rec TD	Max	0.56	0.58	0.59
TE	Receptions	Max	1.34	1.48	1.40
	Rec Yds	RF	16.41	18.69	17.77
	Rec TD	Max	0.64	0.53	0.80
DST	PA	RF	6.73	5.99	7.77
	YdA	RF	61.70	60.50	55.52
	TD	Max	0.85	0.84	0.73
	Sack	SVR	1.18	1.04	1.12
	Int	SVR	0.54	0.74	0.52
	Fum Rec	Max	0.45	0.46	0.62

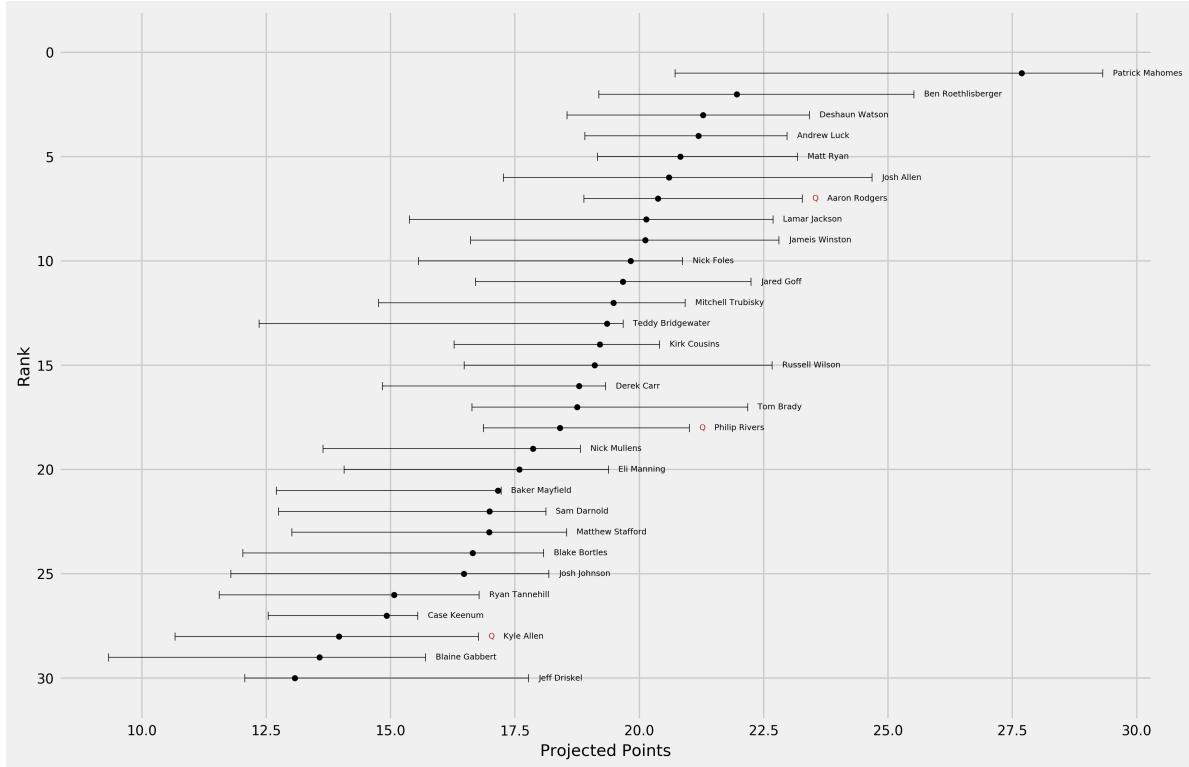
Several of the stats show some signs of difficulty generalizing to the unseen data in the test set. QB - Rush TD, TE - Rec TD, and DST - Fum Rec all had significantly higher test MAE values than their training and cross-validation counterparts. This could provide some future work as these models may require a closer look at data cleaning/thresholding, hyperparameter tuning, or model selection in order to better generalize. However, overall most stat projection test values were nearly equal to or better than their cross-validation MAE value, showing the ability for most models to generalize well to unseen data.

Interestingly, a simple maximum was the most frequent model selection, followed by a random forest, though all machine learning models were utilized for at least one stat. This indicates the largest differences in projecting different stats for different positions. For

example, Rush Yds were found to be optimally projected using RF, OLS, and simple Max for QB, RB, and WR respectively, indicating that the algorithms used by each source are different depending upon not only stat but also position.

Once each player had each of their essential and non-essential stats projected by the optimal model selection, scoring rules from the different types of leagues were applied to reach a final fantasy point projection. For example, FanDuel rewards 0.5 points for each reception, 6 points for each rushing or receiving touchdown, and 0.1 points for every yard gained rushing or receiving. Thus, if a player was projected for 65 rushing yards, 22 receiving yards, 2.2 catches, 0.5 rushing touchdowns, and 0.1 receiving touchdowns, this player would be projected for 13.40 fantasy points in FanDuel’s scoring format.

Once final projections are computed using scoring rules, the projections for any position can be visualized. Figure 10 displays an example visualization for quarterbacks in week 17 scored by FanDuel. In addition to simply showing the generated projections, confidence intervals of each player’s respective floor and ceiling are included as well as player status. Statuses include Q for questionable, IR for players on injured reserved, SSPD for suspended players, and O for players who are out.



**Figure 10:** Example projections for QB week 17. Dots indicate the projections generated using our methodology, with error bars representing total floor and total ceiling for each player. Red text indicates the status of players.

### 3.2 Lineup Optimization

Given optimal player projections, lineups were generated for competition at DraftKings and FanDuel. Both sites have similar lineup constraints, only differing on their salary cap and point system (DraftKings has a 50000 budget and is 1.0 points per reception; FanDuel has a 60000 budget and is 0.5 points per reception). As our goal is to score as many points as possible, projected points were maximized subject to all necessary lineup constraints.

In order to use convex optimization to solve for the optimal lineup, as opposed to brute-force solving for all possible combinations, we define the following problem:

- Let  $N$  be the number of total players.
- Let  $B$  be the salary budget for the contest.
- Let  $w \in \mathbb{R}^N$  be a vector of player-weights, such that  $w_i \in 0, 1$ .
- Let  $P \in \mathbb{R}^N$  be a vector of projected points.
- Let  $S \in \mathbb{R}^N$  be a vector of salaries.
- Let  $D \in \mathbb{R}^{N \times 5}$  be a matrix containing the position of each player, using dummy variables for each position. That is,  $p_{i,j} \in 0, 1$ .
- Let  $C \in \mathbb{R}^{N \times 32}$  be a matrix containing the team of each player, using dummy variables for each team. That is,  $T_{i,j} \in 0, 1$ .

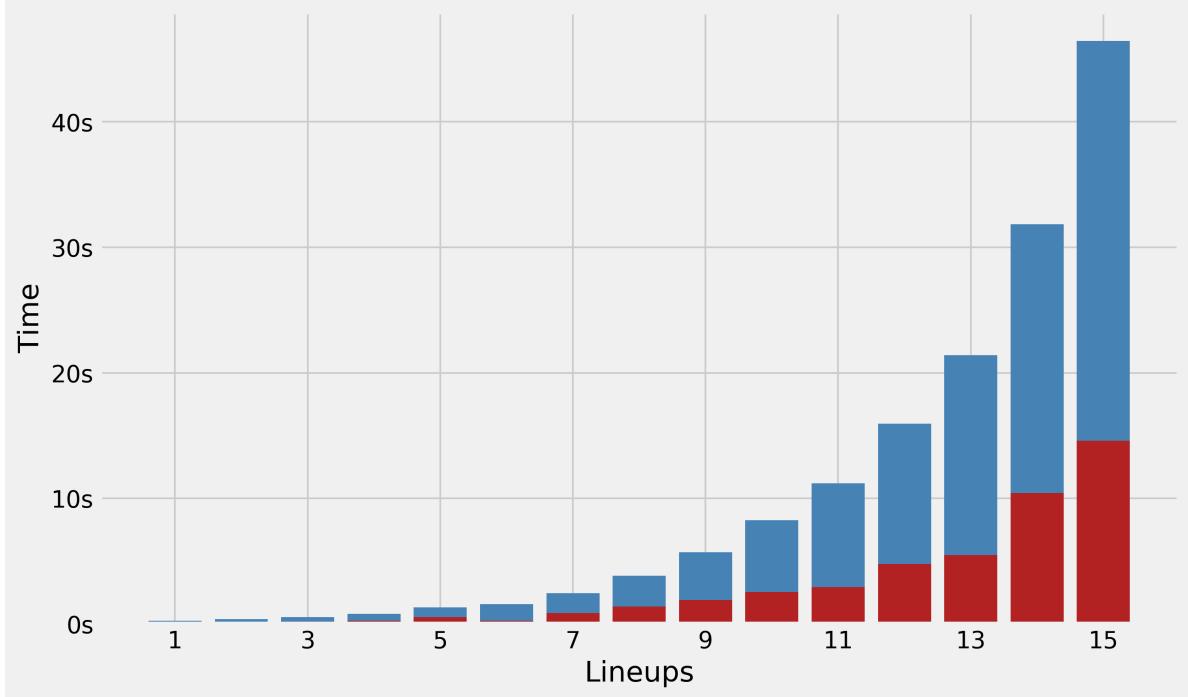
Then, we solve for the  $w$  that solves:

$$\begin{aligned} & \underset{w}{\text{maximize}} && w^T \cdot P \\ & \text{subject to} && w^T \cdot S \leq B, \quad \mathbf{1} \cdot w = 9, \\ & && (w^T \cdot D)_1 = 1, \\ & && 2 \leq (w^T \cdot D)_2 \leq 3, \\ & && 3 \leq (w^T \cdot D)_3 \leq 4, \\ & && 1 \leq (w^T \cdot D)_4 \leq 2, \\ & && (w^T \cdot D)_5 = 1, \\ & && \max_i (w^T \cdot C)_i \leq 4 \end{aligned}$$

where the subscripts  $1, \dots, 5$  on  $(w^T \cdot D)$  correspond to the positions QB, RB, WR, TE, and Defense, respectively. These constraints ensure that each lineup followed the composition rules defined both DFS sites. The lineup optimization routine was written and solved using CVXPY, a Python-embedded modeling language for convex optimization ([14], [15]).

While this formulation worked perfectly when solving for the highest projected lineup, it failed when scaling to the solve for the highest  $n$  projected lineups. As many contests

allow for up to 150 lineup entries per contestant, this limitation must be solved for. Below in figure 11, the exponential nature of the slowdown is shown.

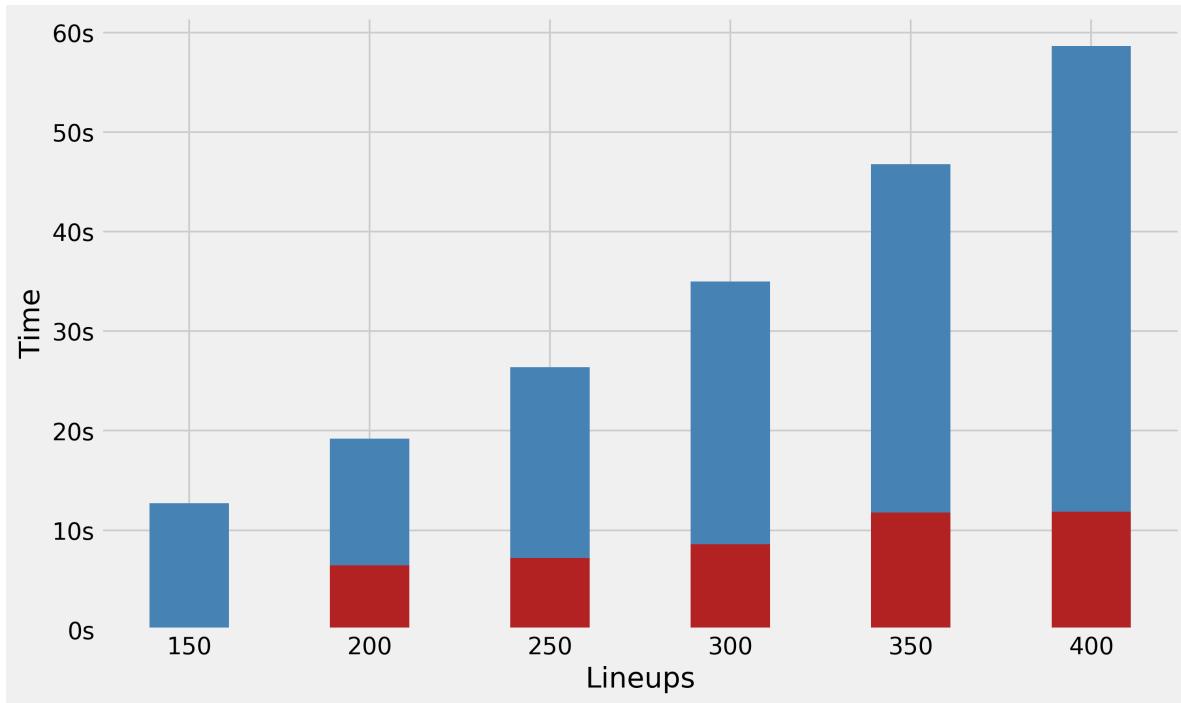


**Figure 11:** Cumulative (blue) and marginal (red) time taken to generate lineups.

As a remedy for this issue, we implement a dropout-like system in which players are dropped from the available pool with probability  $p$  [16]. Thus, the probability that the best possible lineup gets chosen is equal to  $(1 - p)^9$ . Below in Table 5, these probabilities are shown for different values of  $p$ . Using this dropout technique enables us to scale up, almost linearly, the amount of lineups generated, without causing a slowdown. Figure ?? below illustrates the ability for this new technique to scale.

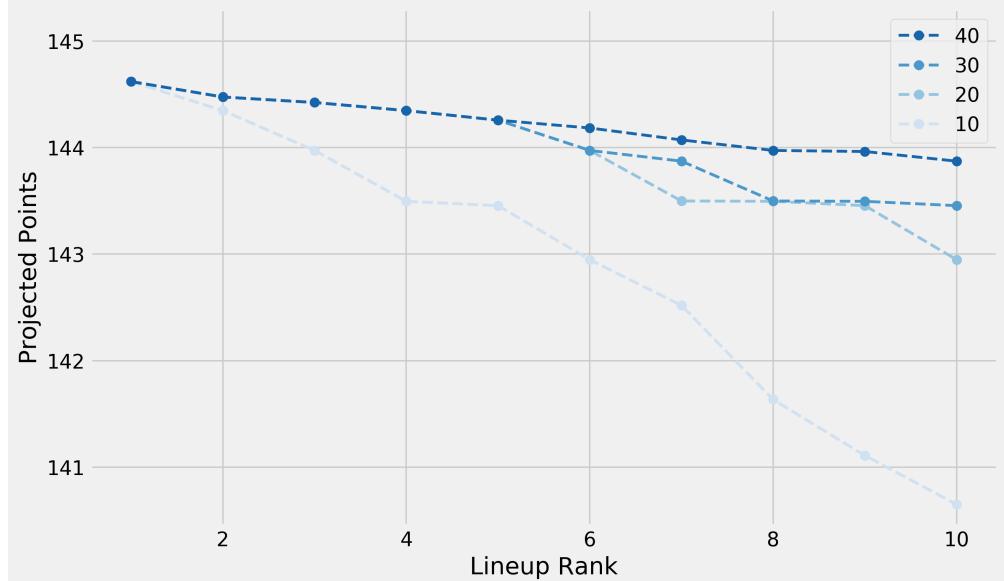
**Table 5:** Probability of best lineup being selected for varying levels of dropout probability  $p$ .

$p$	$\text{Pr}(\text{Max})$
0.00	1.00
0.05	0.63
0.10	0.39
0.15	0.23
0.20	0.13
0.25	0.08
0.30	0.04
0.35	0.02
0.40	0.01

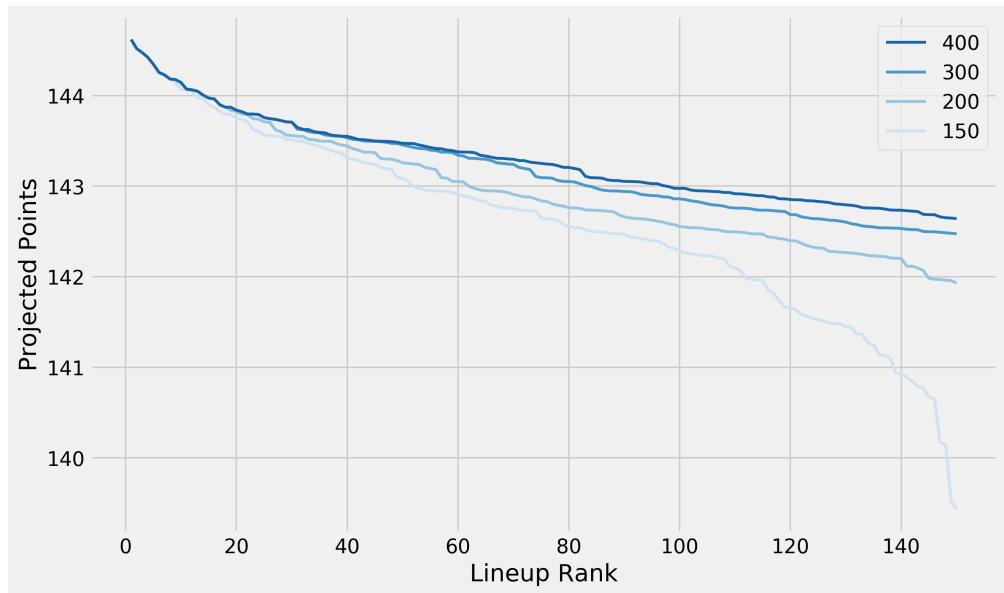


**Figure 12:** Cumulative (blue) and marginal (red) time taken to generate lineups using dropout.

Of course, for what this approach gains in speed, it loses in precision. We have now introduced a non-zero probability that the best possible  $n$  lineups will not be selected if we choose to generate exactly  $n$  lineups, given the randomness of dropout. Using week 1 of 2018 as a test, and  $p$  equal to 25%, 400 lineups were generated, and the first  $n$  lineups generated are examined below in Figure 12. Both the top 10 and top 150 lineups are studied.



(a) Top 10 lineups.



(b) Top 150 lineups.

**Figure 13:** Projected points are plotted against lineup rank for the first  $n$  lineups generated.

From Figure 12, it is clear that if  $n$  lineups are needed, naively choosing  $n$  lineups to generate will produce a sub-optimal output. As 150 lineups are the typical maximum number of allowed lineups for the million dollar contests, we will generate 400, then subset the top 150 for use, using  $p$  equal to 25% as the dropout rate.

To allow further customization of lineups, we introduce a risk parameter  $r \in [-1, 1]$ , which controls the weighting of the projection towards the floor and ceiling values for each player. For example,  $r = 0$  implies that our lineups are optimized for our projections.  $r$

$= 1$  ( $-1$ ) produces lineups that are optimized for the ceiling (floor) projection. Values of  $r$  between  $-1$  and  $0$  use a weighted average of floor and projection for optimization, and similarly, values of  $r$  between  $0$  and  $1$ , a weighted average of projection and ceiling is used. Table 6 below illustrates this in further detail.

**Table 6:** Sample projected point total used in optimization routine for varying levels of  $r$  for a given player.

$r$	Projection Used	Description
-1.0	10.0	Floor
-0.9	10.7	Weighted Average
-0.5	13.5	Weighted Average
0.0	17.0	Actual Projection
0.5	18.5	Weighted Average
0.9	19.7	Weighted Average
1.0	20.0	Ceiling

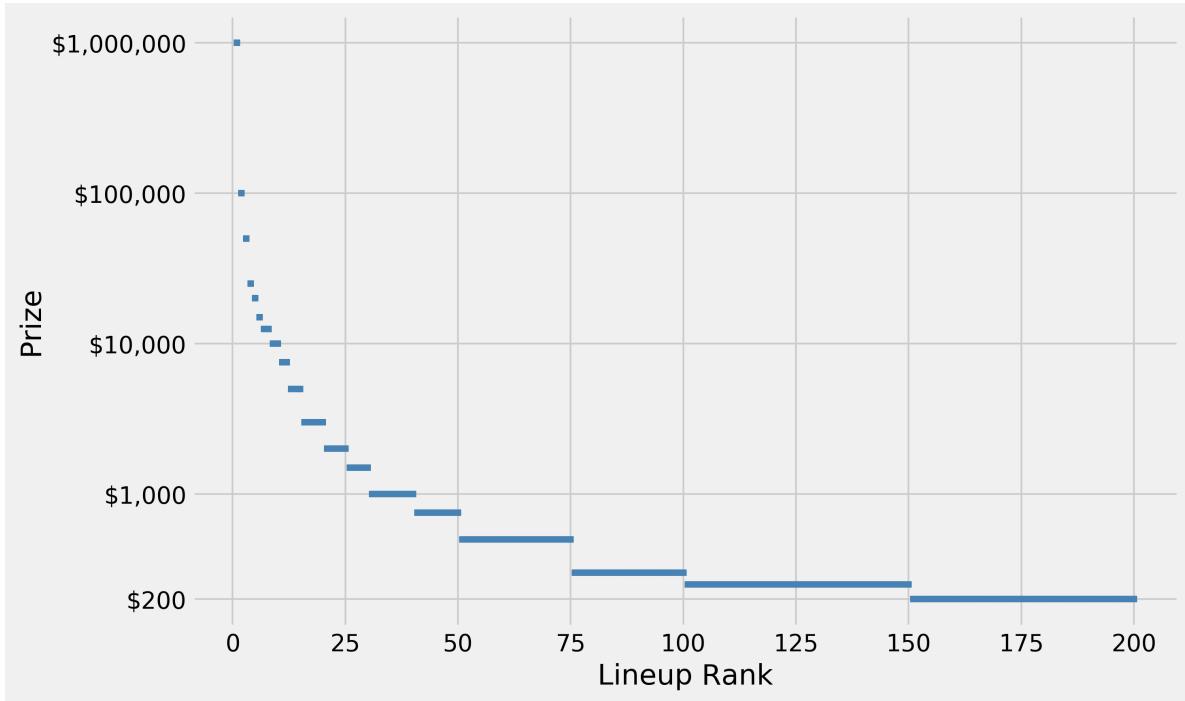
## 4 Results

### 4.1 Contest Info

To test the performance of generated lineups using our projections, lineup results from DraftKings’s main slate (1:00pm and 4:00pm games only), millionaire-maker contests were used. FanDuel contests were not studied due to issues of missing data from multiple weeks. Each of these contests followed a similar buy-in and payout structure. \$1,000,000 was the top prize with a buy-in of around \$20 and a maximum of 150 entries (lineups) per player. Minimum prizes for each contest was typically 50% higher than the buy-in amount. Table 7 below shows details of the contests used in this study. If a lineup finishes in-the-money, the payout structure becomes highly non-linear, showcasing an exponential reward as lineup score increases. Figure 13 illustrates this nature of the payouts by showing the reward for the top 200 scoring lineups in the week 17 contest. Having just one lineup finish in the top 50 will significantly increase the profitability for that week. Table 8 shows the payouts corresponding to all other lineups outside of the top 200.

**Table 7:** DraftKings main slate millionaire-maker contests for the 2018 season with key features shown. These contests were used to measure profitability of generated lineups.

Week	Entry Fee	Max Entries	% Cash	Top Score	Top Prize	Min Prize
1	20	150	23	301.18	1,000,000	30
2	20	150	23	237.84	1,000,000	30
3	20	150	23	251.86	1,000,000	30
4	10	150	21	273.56	1,000,000	15
5	20	150	23	234.36	1,000,000	30
6	10	150	22	245.90	1,000,000	15
7	20	150	23	228.62	1,000,000	30
8	20	150	23	242.80	1,000,000	30
9	20	150	23	245.42	1,000,000	30
10	20	150	23	248.20	1,000,000	30
11	20	150	22	253.72	1,000,000	30
12	20	150	22	252.82	1,000,000	30
13	100	150	22	232.24	1,000,000	150
14	20	150	22	244.30	1,000,000	30
15	20	150	22	206.44	1,000,000	30
16	15	150	20	263.08	1,000,000	25
17	20	150	21	243.22	1,000,000	30



**Figure 14:** Prizes for the top 200 lineups played in the week 17 Millionaire-maker DraftKings contest.

**Table 8:** Payout structure for DraftKings’ week 17 Millionaire-maker contest for lineups placing between below 200.

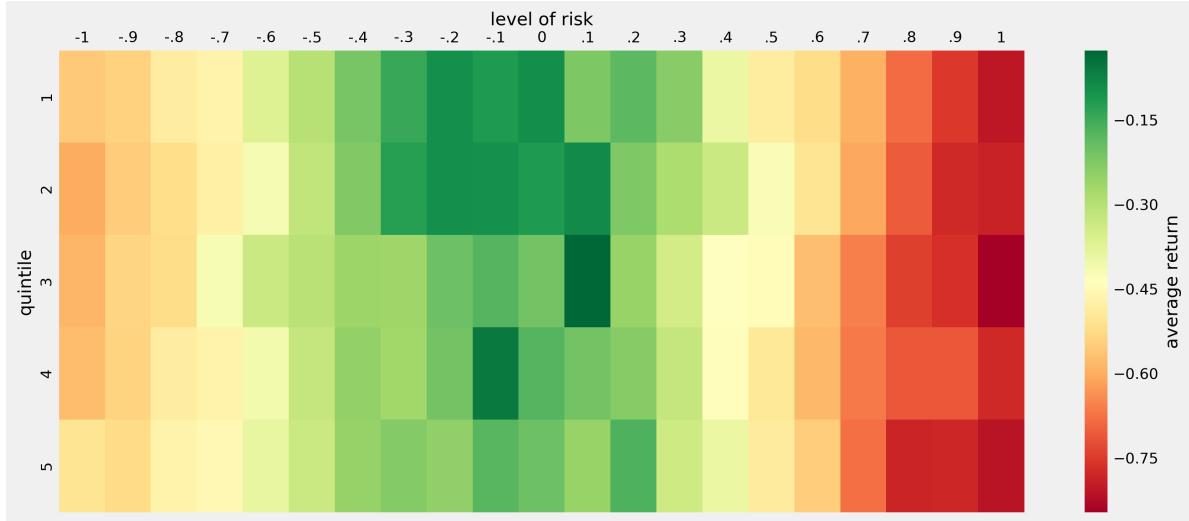
Lineup Rank	Prize (\$)
201st - 300th	\$150
301st - 500th	\$100
501st - 1000th	\$80
1001st - 2000th	\$60
2001st - 5000th	\$50
5001st - 12511th	\$40
12512th - 30413th	\$30
30414th and above	\$0

## 4.2 Training

Weeks 1-14 were used as training to determine the optimal methodology of generating our top 150 lineups. For each week, 150 lineups were generated using 21 varying levels of  $r$  ranging from -1 to 1 (-1, -0.9, ..., 0.9, 1). For each level of  $r$ , lineups were further split into quintiles to examine if higher confidence (projected points) leads to better results. To measure performance, we used the simple average return of each lineup defined below in Equation 3. Prize won for each lineup was determined by taking the actual contest results

and payout structures directly from DraftKings and overlaying our lineups as if they had been entered.

$$Return = \frac{Prize - Entry}{Entry} \quad (3)$$



**Figure 15:** Weeks 1-14 average lineup return is shown for each quintile and each level of  $r$  used.

Figure 14 above details the performance of the lineups tested. While lineups on average lost money, two patterns emerged from these results. First, as the projection used in lineup optimization deviated from our optimal projection (as the level of risk moved away from zero), average return declined, bottoming out at  $r = 1$  (ceiling projection). Second, at around  $r = 0$ , as projected score decreased, so did performance. On average, the bottom quintiles returned less than their respective top quintiles.

### 4.3 Testing

From the results in Weeks 1-14, we define a new methodology for generating the best 150 lineups for competition for use in Weeks 15-17. 150 lineups are generated for  $r = -0.2, -0.1$ , and 0, sorted by projected points. Then we iterate over these three sets of lineups, and successively pick the top available lineup, if not already used, until reaching our maximum allotted amount of 150. In most cases, only the top three quintiles are used, but the exact number varies due to commonality between lineups generated under the three levels of risk. Below in Tables 9 and 10, results are shown for the final three weeks of the season by week and then by quintile.

**Table 9:** Results from weeks 15-17 of our optimal 150 lineups. Average PnL (\$) and average return are show for each week.

Week	PnL	Return
15	1.60	0.08
16	-8.83	-0.59
17	-12.67	-0.63
Average	-6.63	-0.38

**Table 10:** Average PnL (\$) and average return are shown for weeks 15-17 on aggregate for each quintile.

Quintile	PnL	Return
1	-4.89	-0.27
2	-7.06	-0.42
3	-6.94	-0.42
4	-6.83	-0.37
5	-7.44	-0.42

Over the final three weeks of the season, our lineups performed slightly worse, on average, than they did during the first fourteen weeks of the season, only making money during Week 15 action, and losing around 60% over Weeks 16 and 17. Similar to the results from Weeks 1-14, as our projected point total decreased, performance decreased along with it. Table 10 shows the performance of the first quintile outperforms the bottom four quintiles by roughly 15%.

## 5 Future Work

Future work may consist of several improvements to the described methodology. First, more data from other sources may be incorporated. Many additional sources provide similar projections on a weekly basis, but do not store historical projections online. These sources can be scraped each week during the season in order to keep a private history to use for model building. Additionally, several sources are behind paywalls, so paid membership accounts can be created to include them. The law of big data in machine learning states that more data can only result in more accurate predictions, thus including more sources should only improve the projections generated.

Second, several models did not generalize well to the test set. Modulating threshold values and hyperparameter grids may help overcome these issues. Additionally, including other machine learning algorithms such as neural networks may further improve accuracy and precision of generated projections.

Third, during lineup creation, simple linear interpolation between the floor, generated projection, and ceiling are used to create 150 lineups. Using the floor and ceiling levels to instead create a distribution with mode or mean equal to the generated projection and performing Monte Carlo simulations to generate lineups may further improve the ability to generate profitable lineups.

Lastly, tweaks to the lineup optimization constraints may yield better results. Forcing a QB/WR stack, a game stack, or other combinations of players may yield more profitable lineups as performance within an offense or game are often correlated. Limiting the number of appearances each player has among the 150 lineups generated might also improve profitability.

## 6 Conclusion

We have detailed a pipeline for collecting raw stat projections, using machine learning to produce more accurate fantasy point projections, and utilizing convex optimization to generate 150 optimal, unique lineups that satisfy all required constraints from DFS sites. While our methodology failed to yield profitable lineups on average during the training period of Weeks 1-14 and the testing period of Weeks 15-17, we showed that optimizing lineups for floor or ceiling projections are much less profitable than using more accurate projections. Given the exponential payoffs associated with top-ranking lineups each week however, it only takes one excellent performance to easily make up for weeks of losses. Moreover, we have laid the foundation for a functional, and potentially profitable algorithm that can be directly plugged into either DFS site.

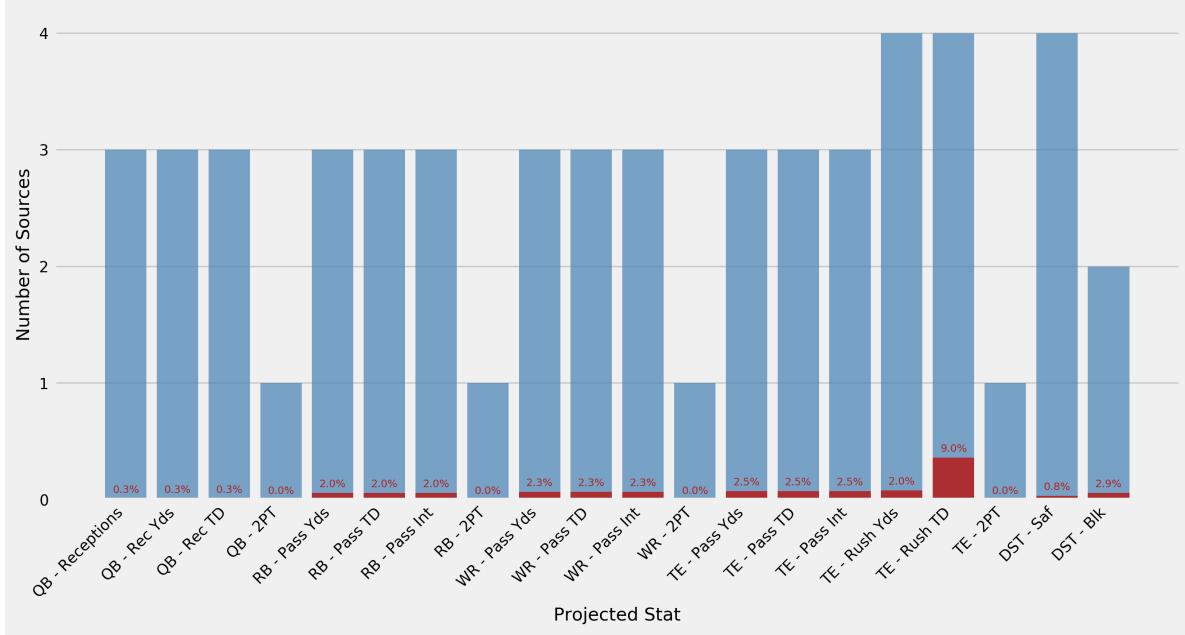
## References

- [1] A. Candee (2019). Nevada Sports Betting Crushes Records With \$5 Billion Year. <https://www.legalsportsreport.com/28482/nevada-sports-betting-revenue-december-2018/>
- [2] P. Everson (2018). Nevada Sports Bettors Wagered Almost \$5 Billion in 2017, Once Again Breaking Betting Records. <https://www.covers.com/Editorial/Article/14f29db1-06c6-11e8-a97f-125516351e3a/Nevada-sports-bettors-wagered-almost-5-billion-in-2017-once-again-breaking-betting-records-bets-sports-betting-odds>
- [3] FSTA. (2019). Press Release: Fantasy Sports Now a \$7 Billion Industry. <https://fstaa.org/press-release-fantasy-sports-now-a-7-billion-industry/>
- [4] A. Rubinsteyn, & S. Feldman (2019). “A Variety of Matrix Completion and Imputation Algorithms Implemented in Python.” <https://github.com/iskandr/fancyimpute>
- [5] H. Zhou & T. Hastie (2005). “Regularization and Variable Selection via the Elastic Net.” *R. Statist. Soc. B*, 67, 30132
- [6] M. Brucher et. al (2019). ElasticNet. [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.ElasticNet.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html)
- [7] T. Chen & C. Guestrin (2016). “XGBoost: A Scalable Tree Boosting System.” *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794
- [8] T. Chen et. al (2019). XGBoost. <https://xgboost.readthedocs.io/en/latest/python/index.html>
- [9] S. Seabold et. al (2019). OLS. [https://www.statsmodels.org/dev/generated/statsmodels.regression.linear\\_model.OLS.html](https://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLS.html)
- [10] A. Liaw & M. Wiener (2002). “Classificationand RegressionbyrandomForest.” *R News*, 2/3:18-22
- [11] M. Brucher et. al (2019). RandomForestRegressor. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>
- [12] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, & V. Vapnik (1997). “Support Vector Regression Machines.” *Advances in Neural Information Processing Systems*, 9, 155161
- [13] M. Brucher et. al (2019). SVR. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html>
- [14] S. Diamond & Stephen Boyd. 2016. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *J. Mach. Learn. Res.* 17, 83 (April 2016), 1-5.

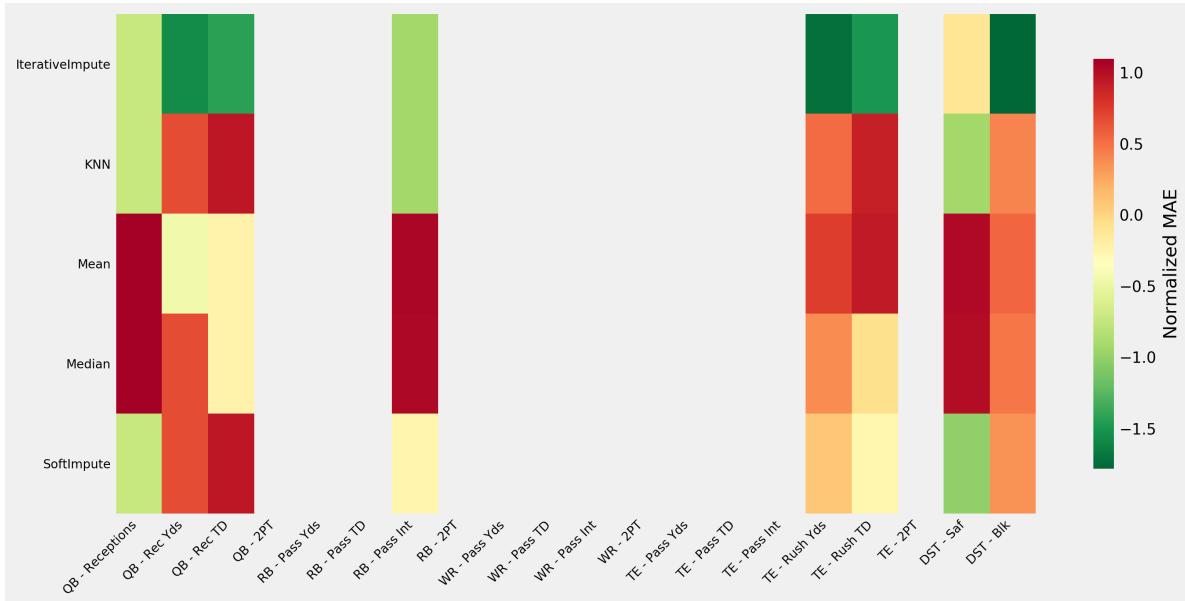
- [15] A. Agrawal et al. 2018. A Rewriting System for Convex Optimization Problems. *Journal of Control and Decision.* 5, 1 (January 2018), 42-60.
- [16] N. Srivastava et al. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (January 2014), 1929-1958.

## 7 Appendix

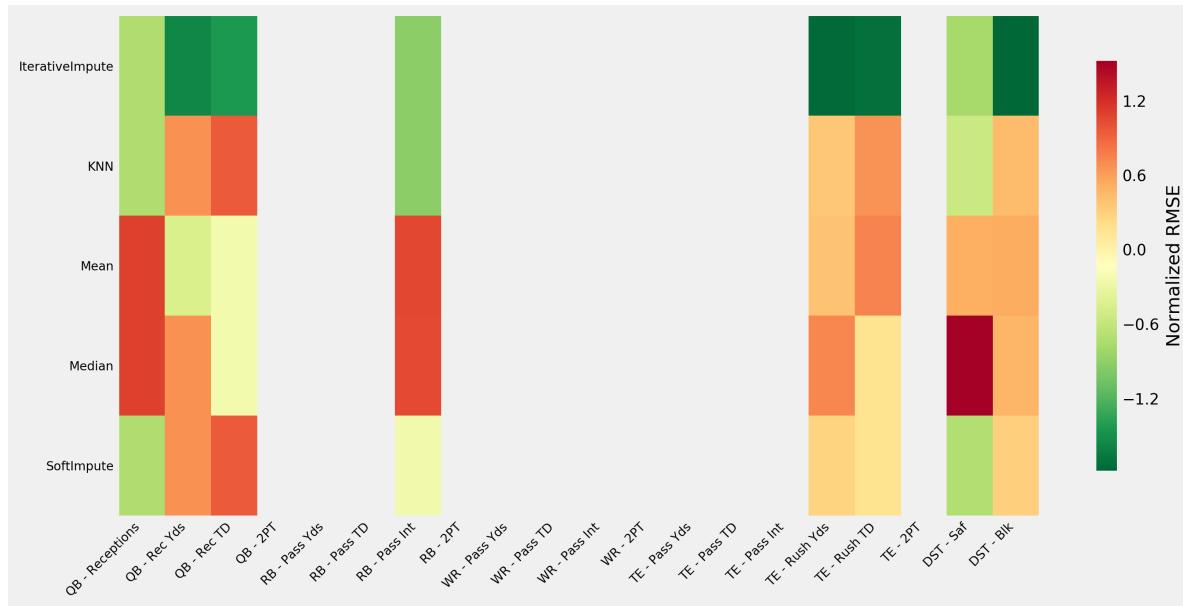
### 7.1 Nonessential Stats Summary



**Figure 16:** Number of sources collected for each nonessential stat (blue). Red indicates percentage of missing data for each respective stat.

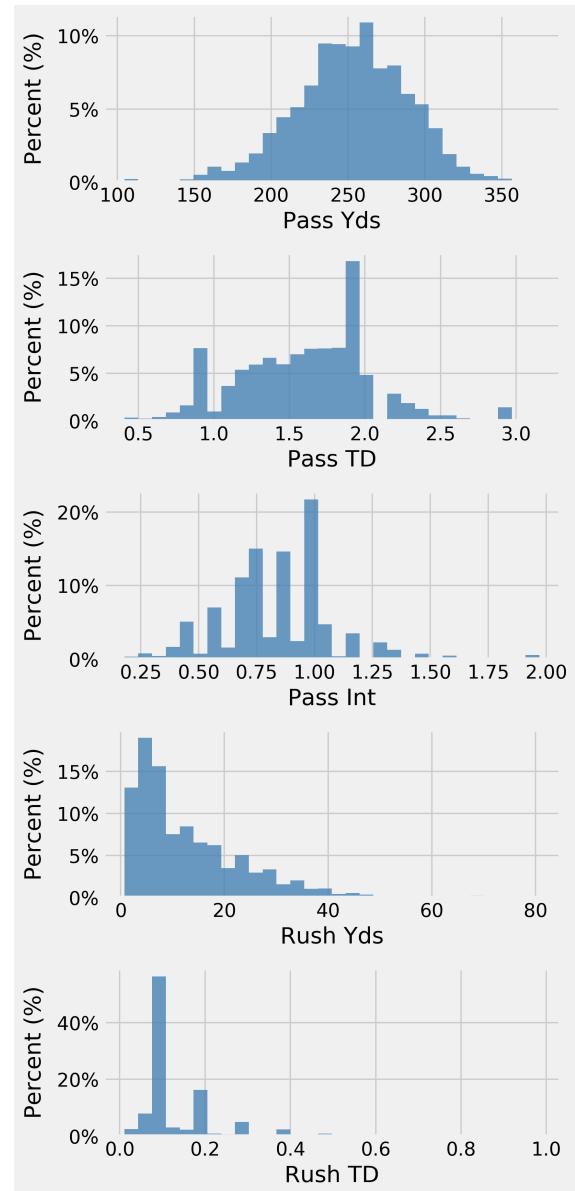
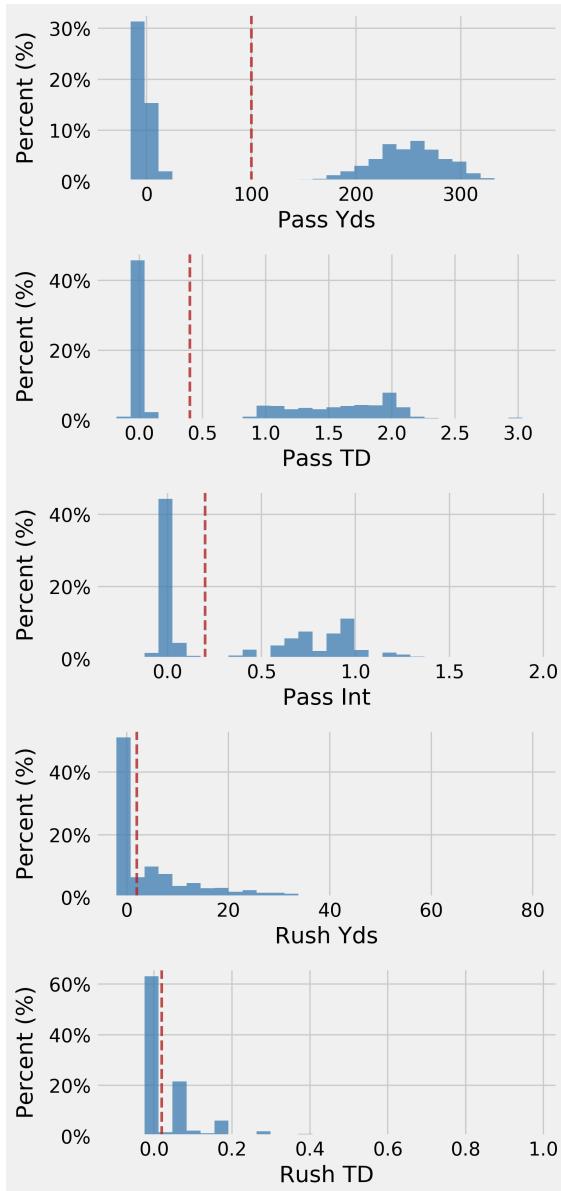


**Figure 17:** Normalized MAE of imputing methods for each nonessential stat.

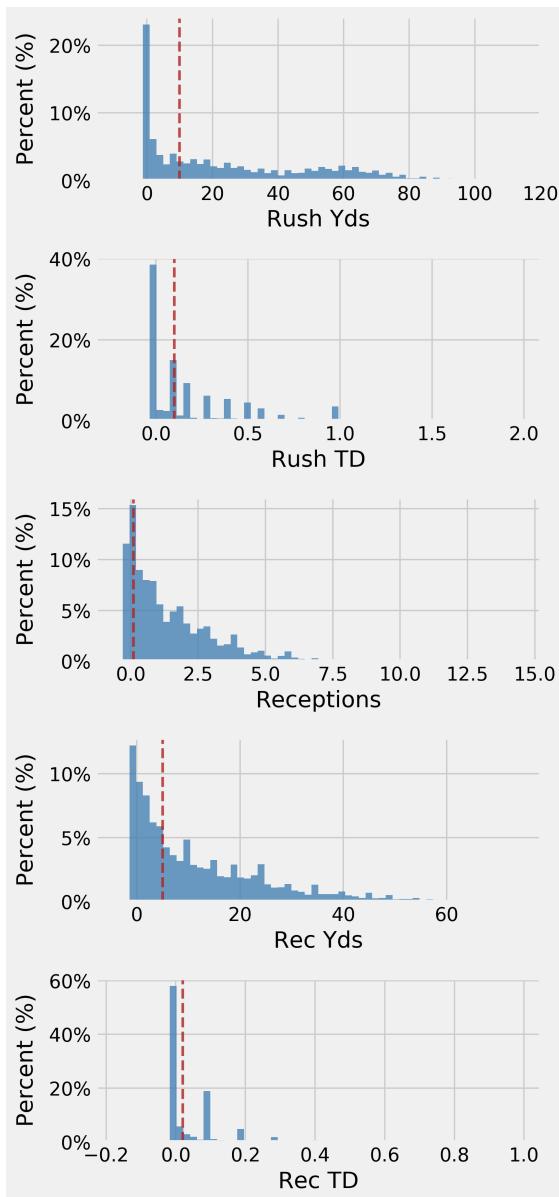


**Figure 18:** Normalized RMSE of imputing methods for each nonessential stat.

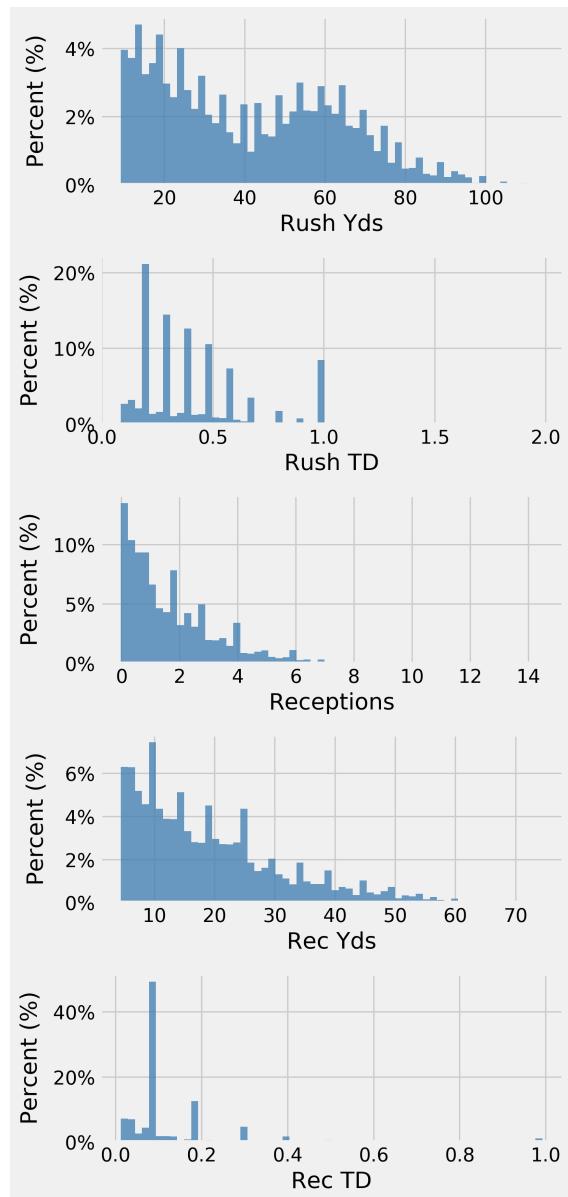
## 7.2 Essential Stats Histograms



**Figure 19:** Essential stat raw histograms and thresholded histograms for QB.

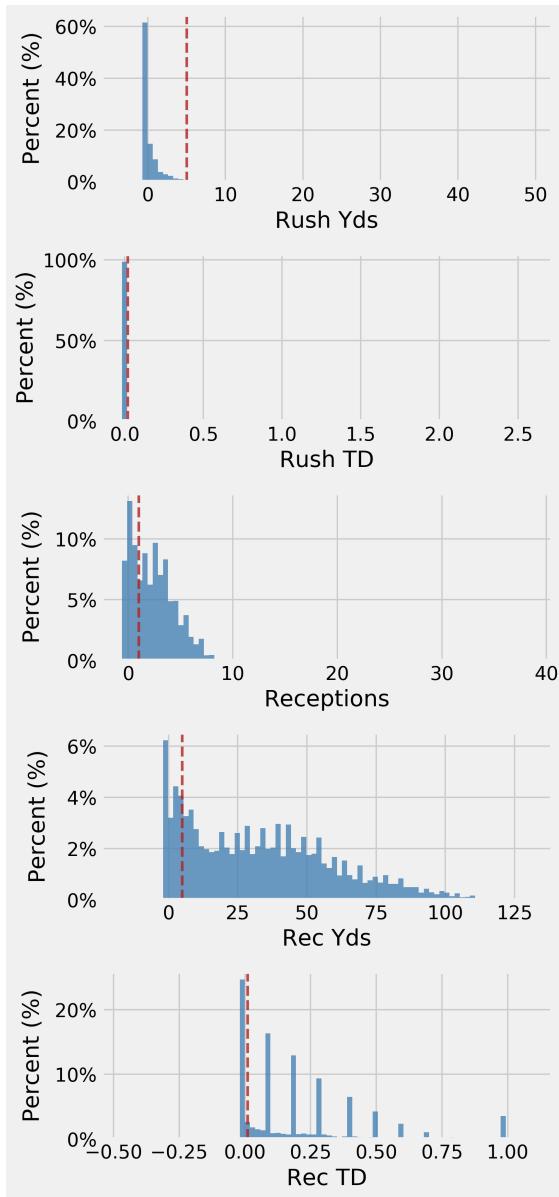


(a) Raw histogram with threshold (red).

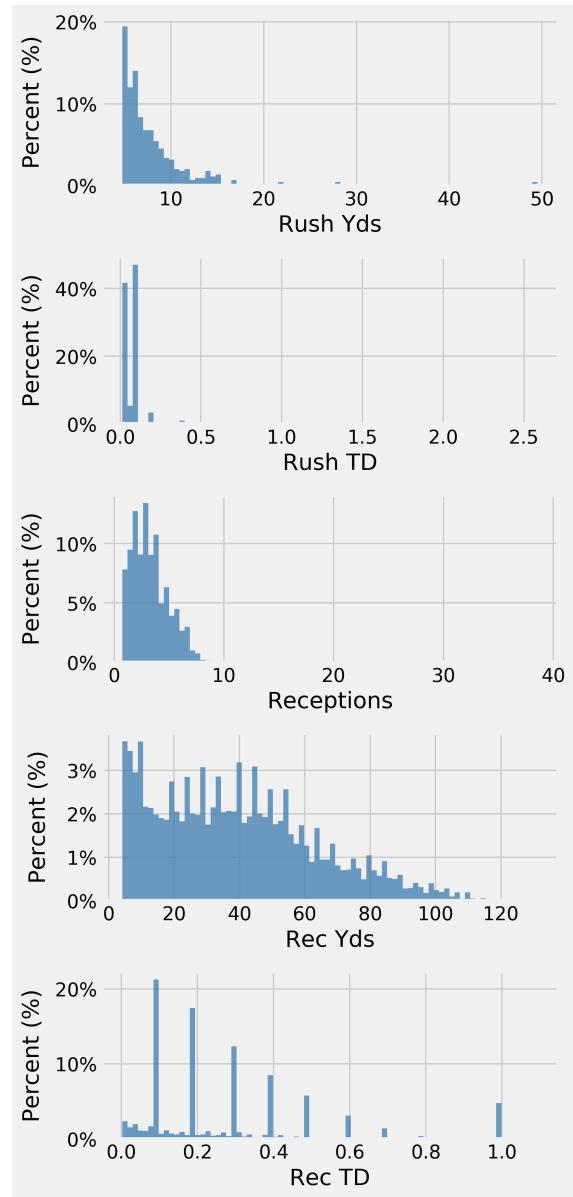


(b) Histogram above threshold.

**Figure 20:** Essential stat raw histograms and thresholded histograms for RB.

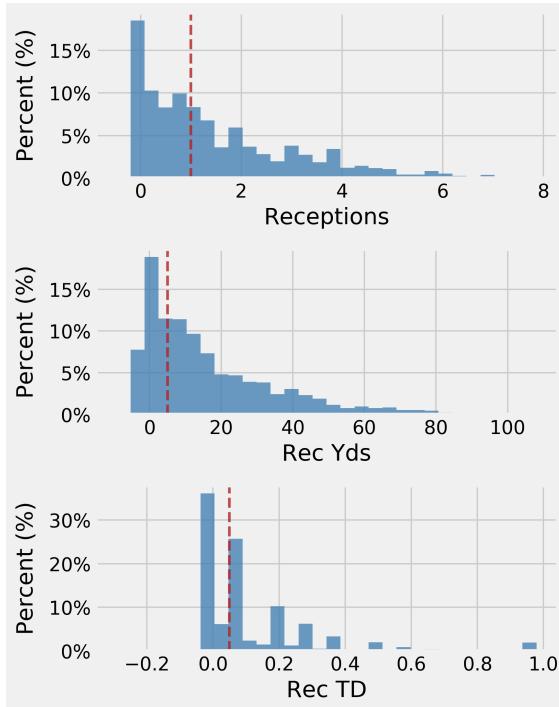


(a) Raw histogram with threshold (red).

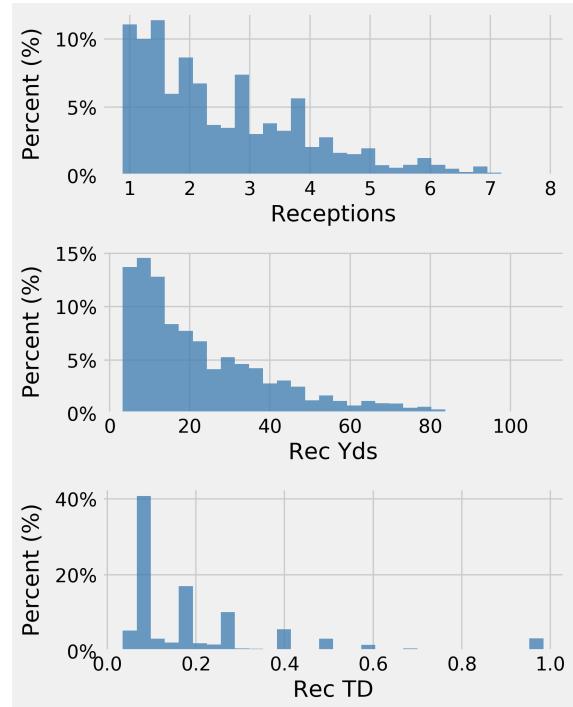


(b) Histogram above threshold.

**Figure 21:** Essential stat raw histograms and thresholded histograms for WR.

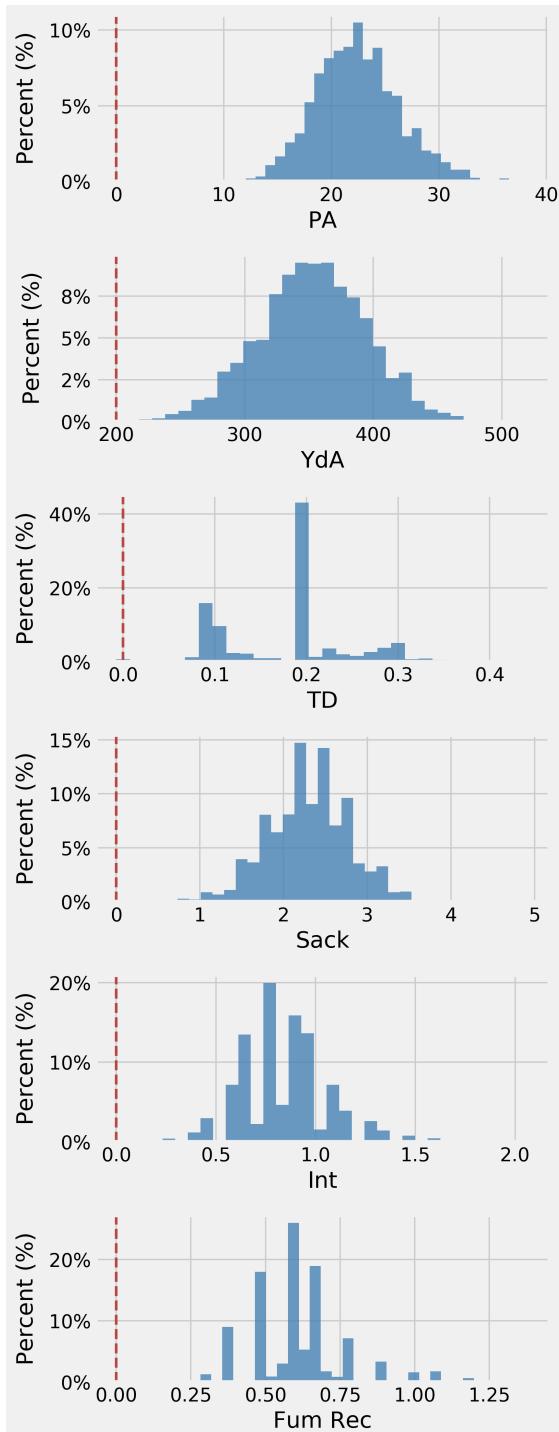


(a) Raw histogram with threshold (red).

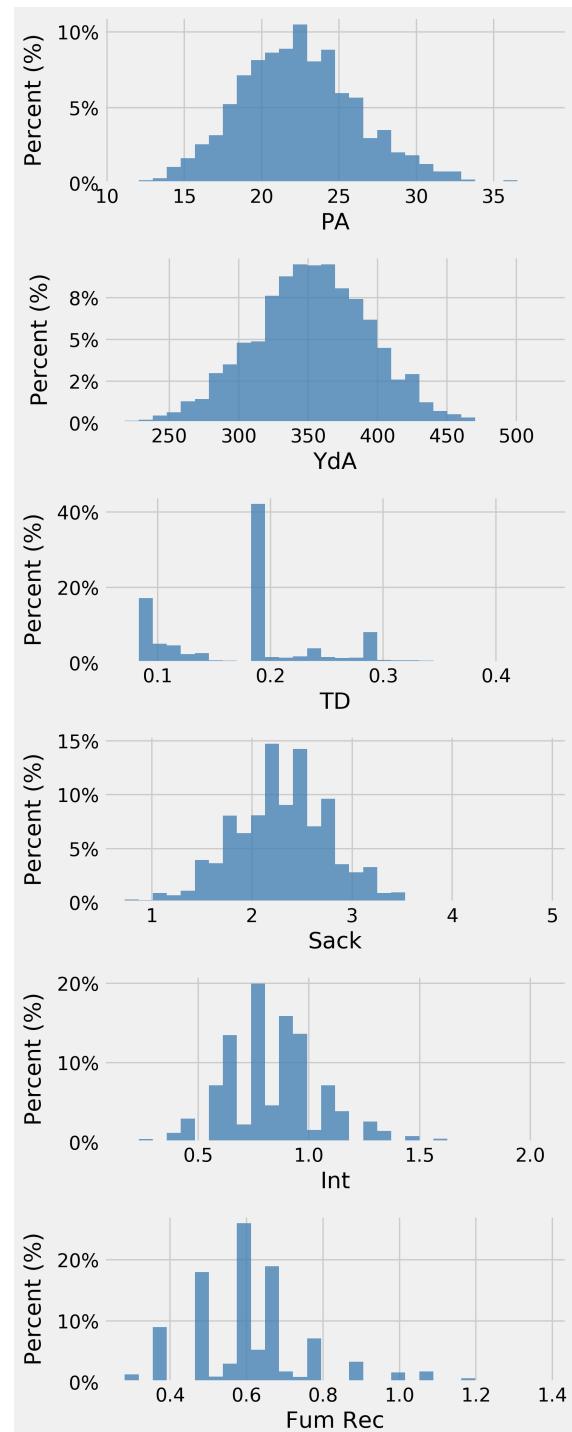


(b) Histogram above threshold.

**Figure 22:** Essential stat raw histograms and thresholded histograms for TE.



(a) Raw histogram with threshold (red).



(b) Histogram above threshold.

**Figure 23:** Essential stat raw histograms and thresholded histograms for DST.

### 7.3 Ensemble Results

**Table 11:** Ordinary least squares (OLS) regression results for ensemble weighting of each source. Asterisks denote statistical significance of regression coefficients, (\*) denoting a p-value less than 0.05, (\*\*) for less than 0.01, and (\*\*\*) for less than 0.001.

Position	Stat	const	CBS	ESPN	FFToday	FantasyPros	NFL	Yahoo
QB	Pass Yds	-40.48	-0.13	0.32	0.81**	-0.45	0.28	0.34
	Pass TD	-0.15	0.46	0.16	0.06	0.64	-0.28	0.08
	Pass Int	-0.01	0.11	-0.24	-0.25	0.42	0.57	0.34
	Rush Yds	3.67***	0.15	-0.02	0.71**	-0.47	0.32	0.1
	Rush TD	0.01	-0.49	0.58	-0.17	0.33	0.87	-0.19
RB	Rush Yds	4.38	0.02	0.44**	-0.16	0.11	0.26	0.23
	Rush TD	0.06	0.27	-0.01	-0.13	0.48	0.25	0.16
	Receptions	0.08	0.12	0.3***	0.26***	-0.05	0.18	0.17***
	Rec Yds	2.51	0.24	0.63***	0.4***	-0.59	0.17	0.09
	Rec TD	0.03	-0.38	-0.37	-0.36**	0.98	0.76***	0.38
WR	Rush Yds	23.23*	0.18	1.47**	-	-2.4	0.21	-0.85
	Receptions	0.23	0.29***	0.45*	0.05	-0.2	0.28***	0.07
	Rec Yds	0.82	-0.03	0.48**	0.11	0.04	0.18	0.21
	Rec TD	0.03	0.03	0.51***	-0.1	0.5	-0.04	0.09
TE	Receptions	0.36***	-0.22	-0.07	0.07	0.45	0.56*	0.08
	Rec Yds	2.62	-0.32	-0.06	-0.29	0.96***	0.61*	0.05
	Rec TD	0.08***	-0.3	0.77***	0.08	-0.39	0.94**	-0.35
DST	PA	0.05	0.31	-	-	0.1	-0.32	0.89**
	YdA	32.28	-0.06	-	-	0.56	-	0.4***
	TD	-0.19	0.15	-0.57	-	-0.13	3.09**	0.38
	Sack	-0.43	0.07	0.08	-	-0.08	0.85	0.37
	Int	-0.3	0.1	-0.42	-	1.25	0.35	-0.06
	Fum Rec	0.52	-0.3	0.62	-	0.04	0.26	-0.46