A Predictive Model for Daily Fantasy Baseball Points

...

By Carlos Beas

Objective

To create a machine learning model that predicts the number of points a baseball player will score for any given gameday in head-to-head (H2H) fantasy league match. The aim is to have a daily indication of which players in a roster are worth keeping in order to maximize chances of winning.

Why?

- As busy professionals who enjoy playing Fantasy Sports, it is often too time consuming a task to stay abreast of all the minutia and details (*on a daily basis!*) needed to be a top ranked player in a fantasy league.
- Head-to-head fantasy games are the simplest way to play.
  - The format is available on all platforms and for almost all sports (Football, Basketball, Baseball...)
  - Requires consistency: Maximizing points while minimizing risk
    - Low bias model is a must. Variance must be optimized to.

Outline

- Concerns and Considerations
- Dataset and Dataflow
- Exploratory Data Analysis
- Feature Selection & Engineering
- Model Testing and Evaluation
- Summary and Conclusions
- Next Steps

Concerns and Considerations

- Size of data set on per-player basis is small
  - Number of records (rows) per player/per year table is on average in the hundreds
    - Propose to merge tables for multiple previous years (<=2015) and use data from 2016 as test set.
  - This also raises concerns on how to deal with NaN. A good strategy is needed to ensure raw data fidelity is maintained.
- Scope of the project has been limited to developing a model for the batting position.
  - Head-to-Head games consist of one pitcher, one catcher, four infielders, and three outfielders: a total of eight hitters and one pitcher.
  - Batter position being used as “guinea pig” to build infrastructure. Then expand to other positions.
- During the development of the model, a sample “batter” player was chosen for which a random forest feature importance table was extracted.
- Assumption is being made that the features extracted for this “batter” player are the same for all batters. Ideally, a production system would derive feature importance on a per-player basis. However, further work must be done to create the code that would automate this process.
- Time window considered for rolling mean was tuned to the three previous games. It was initially set to 10, but it was then revised after noting the fact that series games happen in sets of three against a particular opponent. This improved prediction accuracy significantly.

Dataset and Dataflow

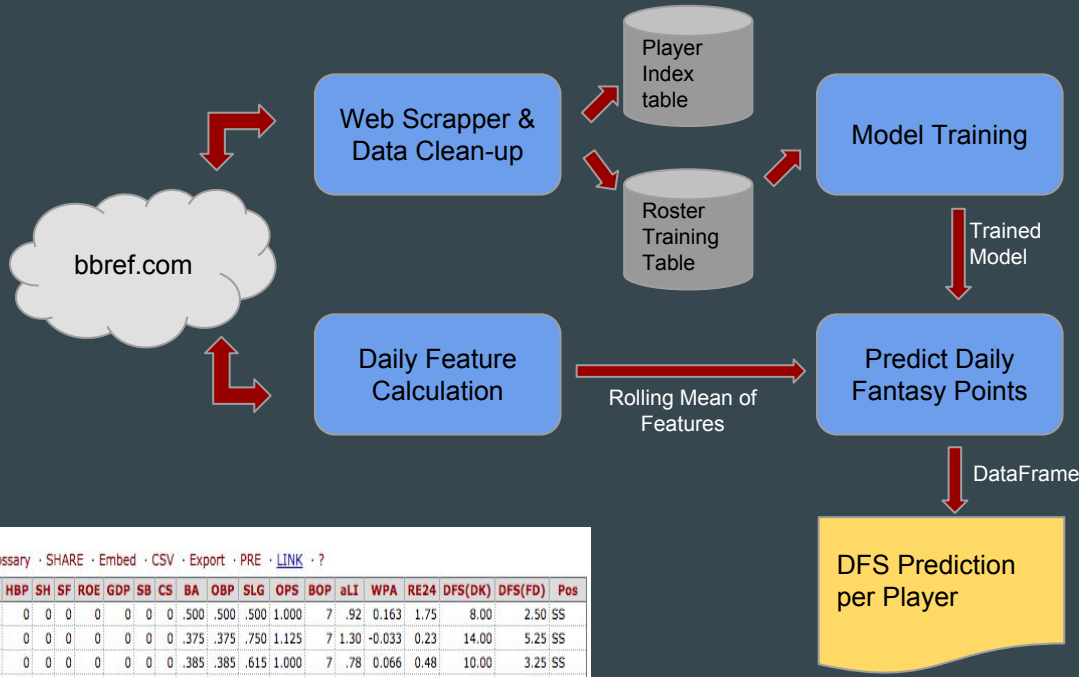
Dataset & Dataflow

Brandon Crawford

Brandon Michael Crawford (twitter: @bcraw35)

Position: Shortstop
Bats: Left, **Throws:** Right
Height: 6' 2", **Weight:** 215 lb.

Born: January 21, 1987 in Mountain View, CA (Age 29.244)
High School: Foothill HS (Pleasanton, CA)
School: University of California, Los Angeles (Los Angeles, CA)
Drafted by the San Francisco Giants in the 4th round of the 2008 amateur draft.
Signed August 14, 2008. (All Transactions)
Debut: May 27, 2011 (Age 24.126, 17,561st in MLB history) vs. MIL 3 AB, 1 H, 1 HR, 4 RBI, 0 SB
Rookie Status: Exceeded rookie limits during 2011 season [*]
Team: Giants 2011-2016
2016 Contract Status: Signed thru 2021, 6 yrs/$75M (16-21) (details) [*]
Service Time (01/2016): 4.094, **Free Agent**: 2022 [*], **Agents**: Wasserman Media Group [*]



2015 Batting Gamelog

Click two rows to sum games (Clear) · More Tricks Below · Glossary · SHARE · Embed · CSV · Export · PRE · LINK · ?

Rk	Gcar	Gtm	Date	Tm		Opp	Rslt	Inngs	PA	AB	R	H	2B	3B	HR	RBI	BB	IBB	SO	HBP	SH	SF	ROE	GDP	SB	CS	BA	OBP	SLG	OPS	BOP	aLI	WPA	RE24	DFS(DK)	DFS(FD)	Pos
1	512	1	Apr 6	SFG	@	ARI	W,5-4	CG	4	4	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	.500	.500	.500	1.000	7	.92	0.163	1.75	8.00	2.50	SS
2	513	2	Apr 7	SFG	@	ARI	L,6-7	CG	4	4	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	.375	.375	.750	1.125	7	1.30	-0.033	0.23	14.00	5.25	SS
3	514	3	Apr 8	SFG	@	ARI	W,5-2	CG	5	5	1	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	.385	.385	.615	1.000	7	.78	0.066	0.48	10.00	3.25	SS
4	515	4	Apr 9	SFG	@	SDP	W,1-0	CG(12)	5	5	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	.278	.278	.444	.722	5	1.40	-0.078	-0.73	2.00	-0.25	SS
5	516	5	Apr 10	SFG	@	SDP	L,0-1	CG	3	3	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0	0	.238	.238	.381	.619	5	2.37	-0.227	-1.73	0.00	-0.75	SS
6	517	6	Apr 11	SFG	@	SDP	L,2-10	GS-4	2	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	.227	.261	.364	.625	8	.29	0.007	0.03	2.00	0.75	SS
7	518	7	Apr 12	SFG	@	SDP	L,4-6	CG	4	4	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	.192	.222	.308	.530	5	1.24	-0.135	-1.28	0.00	-1.00	SS
8	519	8	Apr 13	SFG		COL	L,0-2	CG	4	2	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	.179	.258	.286	.544	8	1.40	0.026	-0.10	4.00	1.50	SS
9	520	9	Apr 14	SFG		COL	L,1-4	CG	4	4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	.156	.229	.250	.479	8	1.23	-0.118	-1.08	0.00	-1.00	SS
10	521	10	Apr 15	SFG		COL	L,2-4	7-8	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.152	.222	.242	.465	9	.34	-0.008	-0.09	0.00	-0.25	PH

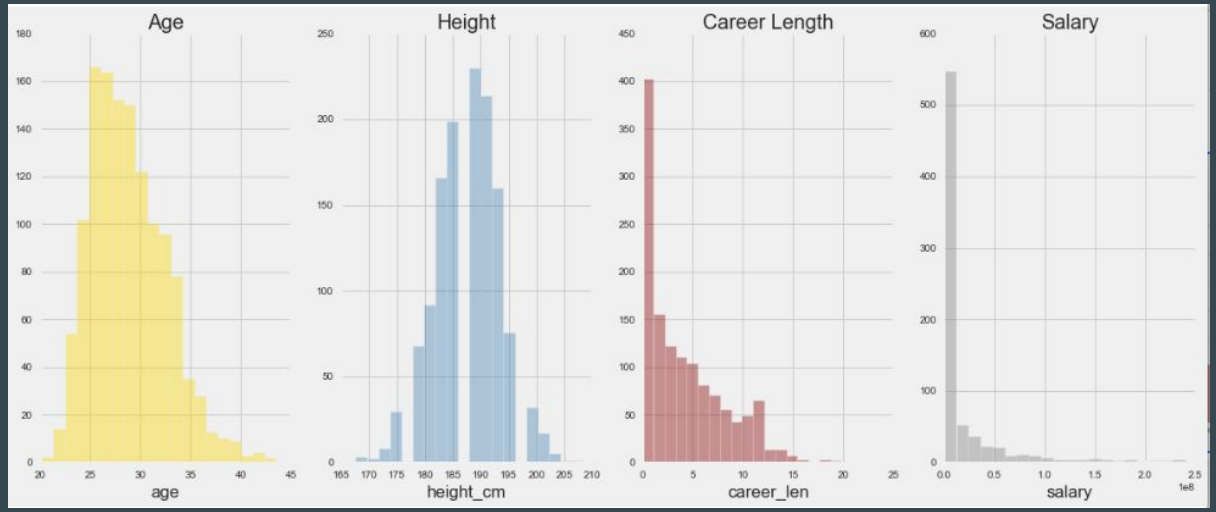
```
active_players_df:  (1304, 14)
batter_players_master_df:  (5980, 43)
```

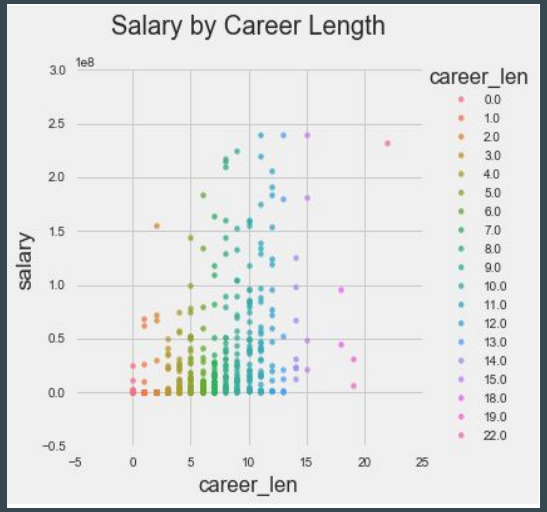
Data Dictionary

Column	Description
DFS(DK)	Daily Fantasy Score/Points (Draft Kings)
RE24	Base-Out Runs Added. Average: RE24=0, above average: RE>0
WPA	Win Probability Added for Offensive Player
H	Hits/Hits Allowed
HR	Home Runs
RBI	Runs Batted In
Opp	Oppossing Team
Opp_Weight	Weight given to opponent teams according to this player's historic performance against such opponent.
Opp_ID	ID given to each oppossing team
away	Flag to indicate if the game was played at player's home stadium or away (@Home=0, away=1)
bat_hand	Flag indicating the batting hand of the player (Right, Left or Both)
throw_hand	Flag indicating the throwing hand of the player (Right, Left or Both)
age	Age of the player
career_len	Career Length of the player
salary	Salary of the player
height	Height of the player
weight	Weight of the player

Exploratory Data Analysis (EDA)

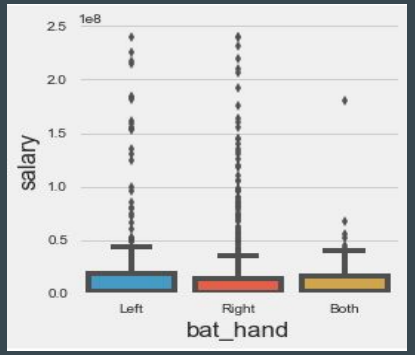
Exploratory Data Analysis



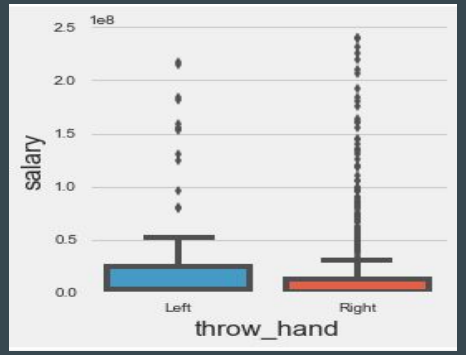


- Obtained data was used to evaluate demographics of Baseball Players
- Key Takeaways:
  - Majority of players in league are between 25 and 35yrs old.
  - There seems to be a preference for players that are 6' or higher in height.
  - Career length seems to be an indicator for salary. Players who make it past 5yrs in the league have better chances of salaries increase.

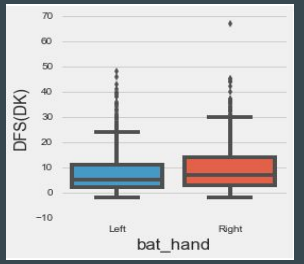
Exploratory Data Analysis



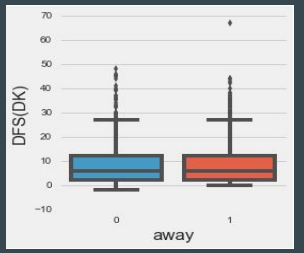
	bat_hand	salary
0	Both	1.420016e+07
1	Left	2.215981e+07
2	Right	1.856929e+07



	throw_hand	salary
0	Left	2.220909e+07
1	Right	1.861097e+07



	bat_hand	DFS(DK)
0	Left	7.462857
1	Right	9.020130



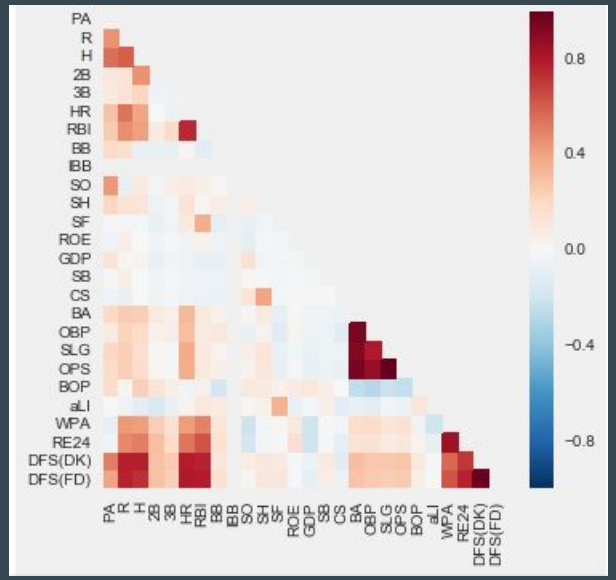
	away	DFS(DK)
0	0	7.971824
1	1	8.276342

- What else can we infer from the data obtained?
  - Salary data is highly skewed towards the right and careful consideration must be taken making any generalization.
  - That being said, when accounting for all players in the distribution, those who are left-handed seem to enjoy a ~18% higher salary than their right-hand counterparts.
  - No significant differences exist for bat_hand or playing away with respect to DFS. Further scrubbing of the data is needed in order to remove outliers.

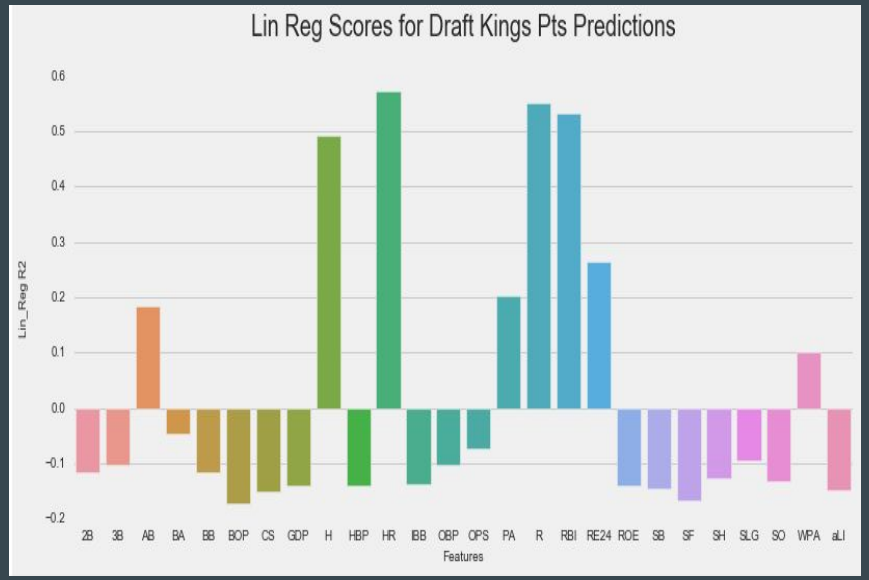
Feature Selection and Engineering

Feature Selection

Correlation Matrix



Linear Regression R2 Score Pareto

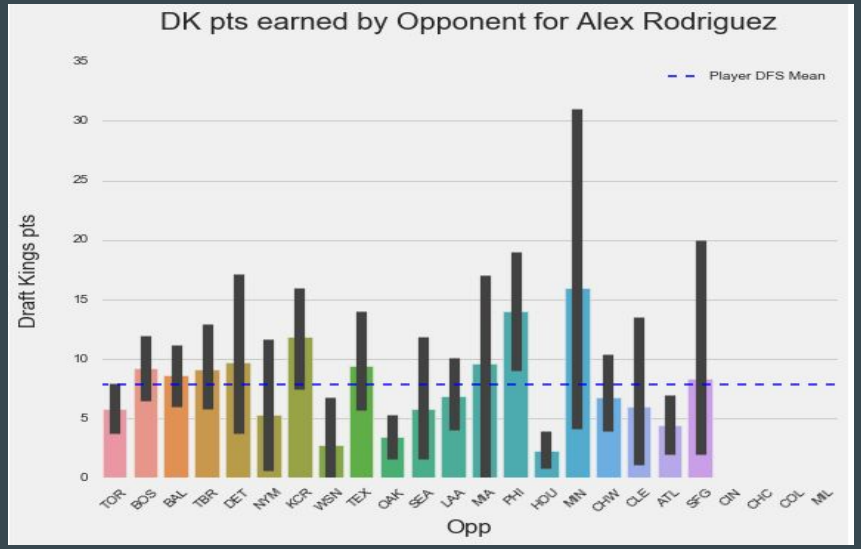


Random Forest Importance Table

	feature	importance
1	RE24_bat	0.155263
10	RBI	0.136844
6	H_bat	0.124735
9	HR_bat	0.117916
2	WPA_bat	0.109441
5	R_bat	0.088813
7	Doubles	0.030869
3	PA	0.025780
43	GSc	0.014444

- A number of sources were considered for feature selection.
- Ultimately, the Random Forest Importance Table was leveraged for feature selection during model training.

Feature Engineering




```
def identify_class(df, opp):
    opponent_class = 0
    #Calculate the mean DFS for this player
    player_DFS_mean = np.mean(df["DK"])
    #Calculate mean DFS per opponent
    DFS_by_opponent = df.groupby("Opp")["DK"].apply(np.mean)
    DFS_by_opponent = DFS_by_opponent.reset_index()
    #What's the mean DFS for this particular opponent?
    opponent_mask = (DFS_by_opponent.Opp == opp)
    opponent_DFS = DFS_by_opponent[opponent_mask]
    DFS = opponent_DFS["DK"].values
    #Classify the opponent according to where it falls
    #with respect to this player's mean DFS score
    try:
        if DFS > player_DFS_mean:
            #Above average
            opponent_class = 1
        elif DFS == player_DFS_mean:
            #Right on the Average
            opponent_class = 0

        elif DFS < player_DFS_mean:
            #Below average
            opponent_class = -1
    except:
        print opp
        opponent_class = 0

    return opponent_class
```

- Hypothesis: Can DFS prediction accuracy be improved if performance against opponent is accounted for in regression model?
- Features “Opp_weight” and “Opp_ID” were created to test this hypothesis...

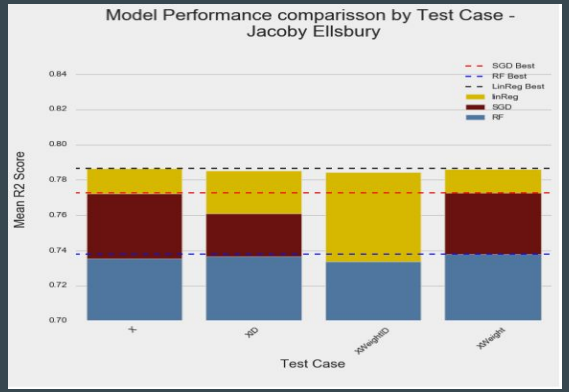
Model Selection

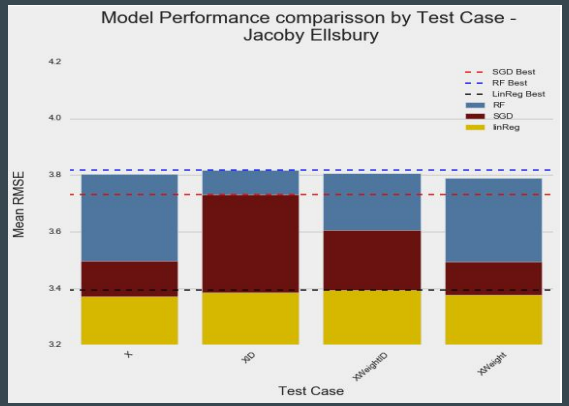
Model Selection and Testing

- Machine Learning models considered:
  - Linear Regression
  - Random Forest Regressor
  - Stochastic Gradient Descent Regressor
- Test Cases Considered for Evaluation:

```
test_cases = {"X":"DK ~ RE24+WPA+H+HR+RBI",
              "XWeight":"DK ~ RE24+WPA+H+HR+RBI+Opp_weight",
              "XID":"DK ~ RE24+WPA+H+HR+RBI+Opp_ID",
             "XWeightID": "DK ~ RE24+WPA+H+HR+RBI+Opp_ID+Opp_weight"
             }
```

- Note:
  - Scenarios were run using data from player in roster showing worst prediction scores

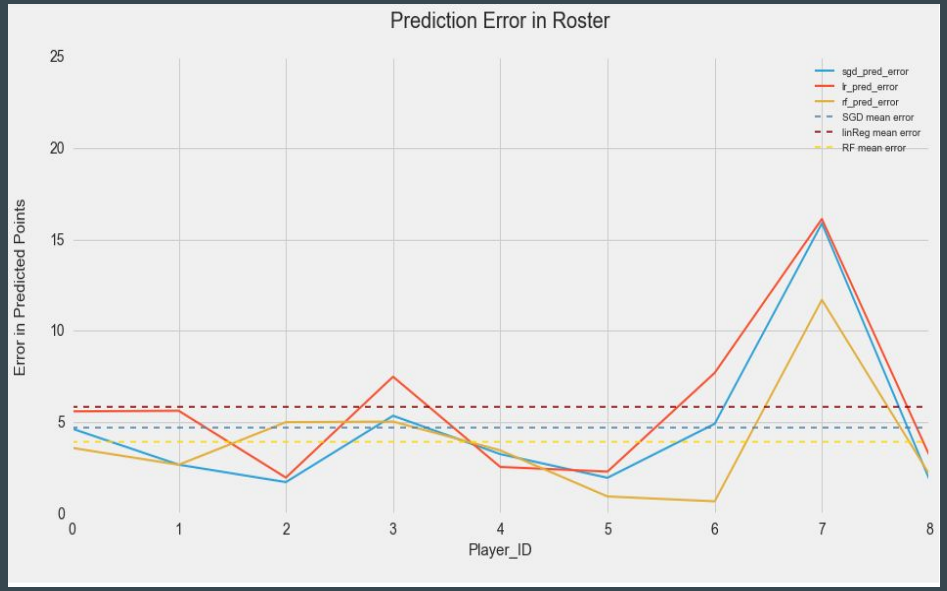




Key Takeaways:

- Linear Regression was found to outperform both Random Forest and Stochastic Gradient Descent during cross-validation scoring.
- No benefit was observed from including Opp_ID as feature.
- Some improvement was observed on Random Forest and SGD performance when including Opp_weight as feature.

Model Evaluation



Random Forest Prediction Error

	player	prediction	actual_score	error
0	David Ortiz	3.593141	0.0	-3.593141
1	Jacoby Ellsbury	3.338390	6.0	2.661610
2	Conor Gillaspie	13.994608	9.0	-4.994608
3	Brandon Crawford	6.973450	12.0	5.026550
4	David Wright	10.546899	14.0	3.453101
5	Yoenis Cespedes	0.928038	0.0	-0.928038
6	Alex Rodriguez	6.338659	7.0	0.661341
7	Mike Trout	2.306130	14.0	11.693870
8	Jose Bautista	2.841466	5.0	2.158534

LinReg Prediction Error

	player	prediction	actual_score	error
0	David Ortiz	5.584894	0.0	5.584894
1	Jacoby Ellsbury	0.380229	6.0	5.619771
2	Conor Gillaspie	7.039736	9.0	1.960264
3	Brandon Crawford	4.515463	12.0	7.484537
4	David Wright	11.456481	14.0	2.543519
5	Yoenis Cespedes	-2.287981	0.0	2.287981
6	Alex Rodriguez	-0.694820	7.0	7.694820
7	Mike Trout	-2.118420	14.0	16.118420
8	Jose Bautista	8.210057	5.0	3.210057

- Models were trained and used to predict last played game per player
- Key Takeaways:
  - Model is able to beat the null accuracy in most cases of out-of-sample prediction. Further data must be collected to draw comprehensive conclusion
  - Linear regression model was found to have the highest mean error when used for actual predictions.
  - Random Forest regressor displays significantly better performance in predictions using unseen data.
    - This is attributed to its superior ability to cope with variance in the data while maintaining low bias.

Summary and Conclusions

- Summary:
  - Python (Pandas) was used to scrape data from online sources to build the dataset used to train prediction models and make predictions.
  - EDA was performed on the collected data to formulate hypotheses and perform feature selection and engineering.
  - Three different machine learning models were considered for the task of making DFS predictions: Linear Regression, Random Forest and Stochastic Gradient Descent. Additionally, multiple test cases were considered to evaluate sources of improvement.
- **Conclusions:**
  - While linear regression was found to outperform Random Forest and SGD during training and cross-validation scoring, this model is not able to handle high variance from actual prediction from unseen data when compared to its counterparts.
  - Random Forest was instead found to be the best model for making predictions when using rolling mean of prediction features.
  - Additionally, including Opp_weight was found to provide an additional benefit, albeit small.
  - Model needs improvement before it can be used with confidence for point prediction accuracy.

Next Steps

- Evaluate prediction accuracy when using a more elaborate rolling mean algorithm.
    - Give more importance to recent games than past ones. Perhaps using a logarithmic rolling mean that weighs recent games more heavily.
- Consider further feature engineering based on salary data discussed in EDA section of the work.
- Deploy as a Flask application.
- Use 2016 season data for training now that season is over... and document model improvements due to this change.
- Build-upon existing model to expand to explore its functionality to the task of roster optimization.
- Evaluate performance using other types of Machine Learning models: Naive Bayes, SVM, Neural Networks.

Back Up

Random Forest Regressor Performance

	R2	RMSE	mean_score	percentage_error	player
0	0.814723	2.589004	6.148315	0.421092	Brandon Crawford
1	0.803140	2.909661	8.811518	0.330211	Joey Votto
2	0.727957	3.854270	8.535533	0.451556	Jacoby Ellsbury
3	0.798878	3.113200	7.968310	0.390698	David Wright
4	0.819033	3.413558	10.192389	0.334912	Mike Trout
5	0.822338	2.949185	7.866667	0.374896	Alex Rodriguez
6	0.883690	2.784371	9.352113	0.297726	Jose Bautista
7	0.776632	3.031747	6.537931	0.463717	Carl Crawford
8	0.899672	2.546388	8.922353	0.285394	David Ortiz
9	0.853047	3.139545	8.466238	0.370831	Yoenis Cespedes
10	0.820375	2.251168	5.383481	0.418162	Conor Gillaspie

Test Case Scores - Random Forest

```
rf_scores_df = pd.DataFrame()
rf_scores_df["test"] = rf_scores.keys()
rf_scores_df["R2"] = rf_scores.values()
rf_scores_df["RMSE"] = rf_RMSE.values()
rf_scores_df
```

	test	R2	RMSE
0	X	0.735329	3.802699
1	XID	0.736785	3.817321
2	XWeightID	0.733728	3.806746
3	XWeight	0.737741	3.788775

- Results show that including Opp_weight as feature can lead to benefits in prediction accuracy albeit minimum (0.002 improvement).

Gridsearch Parameters for SGD Regressor

```
best estimator SGDRegressor(alpha=0.01, average=False, epsilon=0.1, eta0=0.01,
       fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling',
       loss='squared_loss', n_iter=5, penalty='l1', power_t=0.25,
       random_state=None, shuffle=True, verbose=0, warm_start=False)

==========

best parameters {'penalty': 'l1', 'alpha': 0.01, 'loss': 'squared_loss'}

==========

best score 0.781526186917
```

Gridsearch Parameters for Random Forest Regressor

```
best estimator RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=4,
           max_features=None, max_leaf_nodes=8, min_samples_leaf=1,
           min_samples_split=4, min_weight_fraction_leaf=0.0,
           n_estimators=200, n_jobs=1, oob_score=False, random_state=None,
           verbose=0, warm_start=False)

==========

best parameters {'max_features': None, 'max_leaf_nodes': 8, 'min_samples_split': 4, 'n_estimators': 200, 'max_depth':
 4}

==========

best score 0.749069576701
Cross Validated Scores:  [ 0.56193825  0.85161016  0.7899936   0.76968003  0.74717395  0.82656001
  0.76536796  0.58133955  0.74648944  0.72038767]
Average Score:  0.736054062554
```