Predicting Fantasy Football Player Performance

BY: LAUREN SHARESHIAN

Motivation: Slaughter My Fantasy Football Opponents

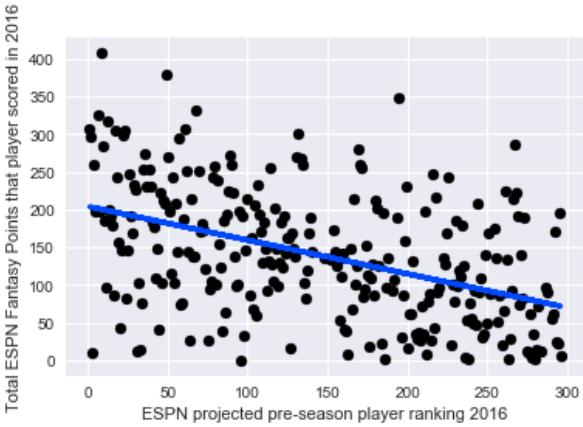
Each pre-season, ESPN puts out projected rankings for how each player will perform that season:

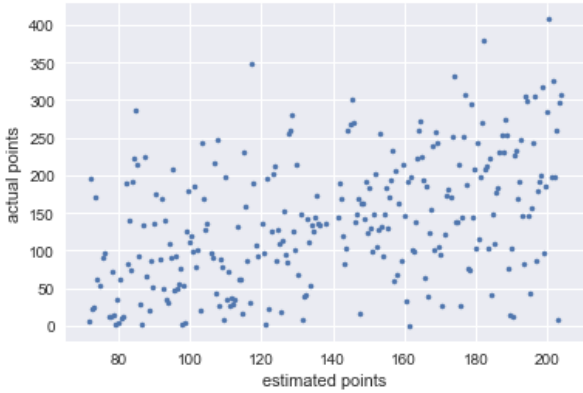
	position	espn_projected_ranking	actual_points_scored
Antonio Brown	WR	1	307.3
Odell Beckham Jr.	WR	2	296.6
Adrian Peterson	RB	3	9.0
Julio Jones	WR	4	259.9
Todd Gurley	RB	5	198.2
Ezekiel Elliott	RB	6	325.4
DeAndre Hopkins	WR	7	197.4
David Johnson	RB	8	407.8
Devonta Freeman	RB	9	284.1
A.J. Green	WR	10	186.4

Who actually were the leading scorers?

	position	espn_projected_ranking	actual_points_scored
David Johnson	RB	8	407.8
Aaron Rodgers	QB	49	380.0
Matt Ryan	QB	195	347.5
Drew Brees	QB	68	332.3
Ezekiel Elliott	RB	6	325.4
Le'Veon Bell	RB	12	317.4
Andrew Luck	QB	61	307.7
Antonio Brown	WR	1	307.3
Jordy Nelson	WR	17	304.7
Mike Evans	WR	23	304.1

How well did ESPN do?





A linear model had the lowest MSE:

```
Degree: 0 MSE: 7355.96861681
Degree: 1 MSE: 5911.95439097
Degree: 2 MSE: 5992.70389363
Degree: 3 MSE: 6024.60327625
Degree: 4 MSE: 6068.68277189
```

But the adjusted R^2 using Ridge CV with 5 folds and normalization was only:

```
adj r2 train = 0.265554329317
adj r2 test = 0.157392678893
```

How well did ESPN do?

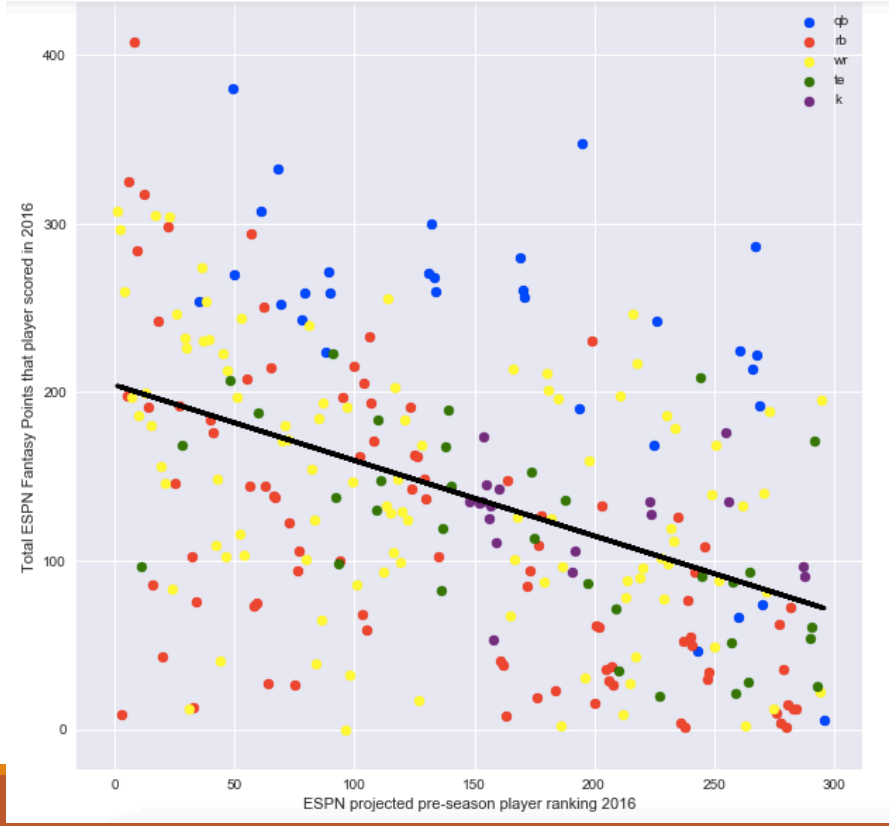
Most underestimated players:

	projection	points	lr_predicted_points	residual
Matt Ryan	195	347.5	117.019916	230.480084
David Johnson	8	407.8	200.640539	207.159461
Dak Prescott	267	286.9	84.823740	202.076260
Aaron Rodgers	49	380.0	182.306606	197.693394
Drew Brees	68	332.3	173.810393	158.489607
Kirk Cousins	132	300.3	145.191570	155.108430
Matthew Stafford	169	279.8	128.646313	151.153687
Joe Flacco	226	242.5	103.157674	139.342326
Davante Adams	216	246.7	107.629365	139.070635
Sam Bradford	268	221.9	84.376571	137.523429

Most overestimated players:

	projection	points	lr_predicted_points	residual
Adrian Peterson	3	9.0	202.876385	-193.876385
Keenan Allen	31	12.3	190.355650	-178.055650
Jamaal Charles	33	13.4	189.461311	-176.061311
Josh Gordon	96	0.0	161.289658	-161.289658
Eddie Lacy	20	42.8	195.274510	-152.474510
Danny Woodhead	64	27.1	175.599069	-148.499069
Eric Decker	44	40.4	184.542451	-144.142451
Ameer Abdullah	75	26.8	170.680209	-143.880209
Markus Wheaton	127	17.1	147.427415	-130.327415
Vincent Jackson	98	32.3	160.395319	-128.095319

Are there other features that would increase predictiveness? Probably position…



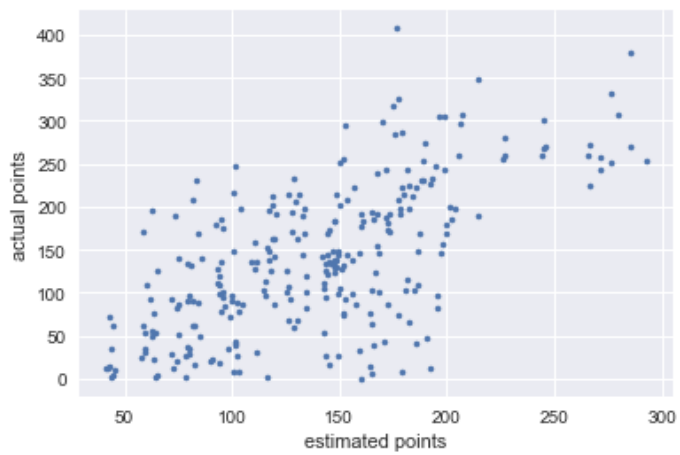
Position is a significant feature!

Without Including Position:

```
adj r2 train = 0.265554329317
adj r2 test = 0.157392678893
```

Including Position:

```
adj r2 train = 0.394117973532
adj r2 test = 0.284009707081
```

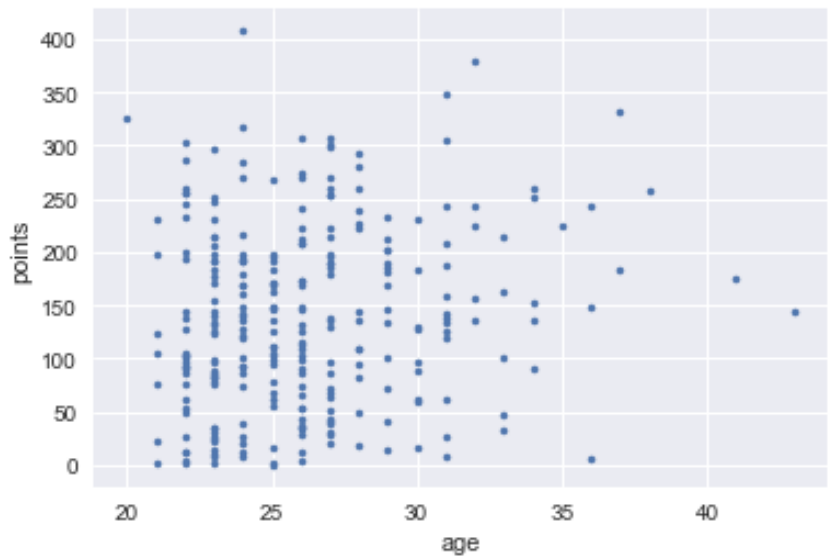


What about other factors? Age is probably an important one.

PLAYER	TEAM	NO	POS	EXP	HEIGHT	WEIGHT	AGE	SCHOOL
Aaitui, Isaako	TEN	71	DT	2	6'4"	307	29	UNLV
Abbrederis, Jared	DET	-	WR	3	6'1"	195	25	Wisconsin
Abdesmad, Mehdi	TEN	92	DE	0	6'6"	284	24	Boston College
Abdul-Quddus, Isa	MIA	24	S	6	6'1"	196	26	Fordham
Abdullah, Ameer	DET	21	RB	2	5'9"	203	23	Nebraska

http://www.foxsports.com/nfl/players?teamId=0&season=2016&position=0&page=1&country=0&grouping=0&weightclass=0

Crap…age distribution actually looks like this:




```
age_matrix.corr()['points'].sort_values(ascending = False)
```

```
points            1.000000
position[T.QB]    0.392798
Age               0.154122
position[T.WR]    0.036410
position[T.TE]   -0.106538
position[T.RB]   -0.205405
projection       -0.452122
```

Without Including Age:

```
adj r2 train = 0.394117973532
adj r2 test = 0.284009707081
```

Including Age:

```
adj r2 train = 0.392884064894
adj r2 test = 0.26935044523
```

Other features? How about team?

Certain teams might improve your points:

```
Team[T.NO]        0.150789
Team[T.ATL]       0.133506
Team[T.OAK]       0.069395
```

Unfortunately, team made the adjusted R^2 worse.

Certain teams might decrease your points:

```
Team[T.SF]       -0.113889
Team[T.CHI]      -0.089097
Team[T.LA]       -0.092878
```

What other factors can we take into account? Players' colleges. Consider two different measures:

1: Historic Powerhouse Schools:

	School	School_Championships
1	Alabama	11
2	Notre Dame	8
3	Oklahoma	7
4	USC	7
5	Ohio State	6

#2: How many current top 300 players are from each school:

	School	Num_of_players_from_school
0	Miami (FL)	10
1	Ohio State	7
2	California	7
3	Alabama	7
4	Clemson	7

Neither was a significant factor.

Source: https://en.wikipedia.org/wiki/College_football_national_championships_in_NCAA_Division_I_FBS

Other Factors? Depth Chart

THOMAS RAWLS	RB	1	SEA
CHRISTINE MICHAEL	RB	2	SEA
CJ PROSISE	RB	3	SEA
ALEX COLLINS	RB	4	SEA
ZAC BROOKS	RB	5	SEA

To merge with depth chart, I needed to delete these five players that weren't listed on the depth chart, reducing our data set from 277 to 272:

```
1 Andre Johnson
2 Anquan Boldin
3 Arian Foster
4 Reggie Bush
5 Ryan Fitzpatrick
```

Unfortunately, depth decreased our adjusted R^2 from 28.4 % to 26.9%

Source: (thank you, random internet, dude!)
https://www.reddit.com/r/nfl/comments/4n2uzj/2016_nfl_depth_charts_all_teams_all_positions/

What if I wanted to be an independent woman and not use ESPN at all?

	points	Intercept	position[T.QB]	position[T.RB]	position[T.TE]	position[T.WR]	2015_points
Cam Newton	254.3	1.0	1.0	0.0	0.0	0.0	391
Tom Brady	258.6	1.0	1.0	0.0	0.0	0.0	344
Russell Wilson	270.1	1.0	1.0	0.0	0.0	0.0	342
Blake Bortles	271.1	1.0	1.0	0.0	0.0	0.0	324
Carson Palmer	243.1	1.0	1.0	0.0	0.0	0.0	309

With just ESPN rankings:

```
adj r2 train = 0.265554329317
adj r2 test = 0.157392678893
```

With just last years' points and positions:

```
adj r2 train = 0.340115683614
adj r2 test = 0.19374638751
```

Final Performance Summary

With just ESPN rankings:

```
adj r2 train = 0.265554329317
adj r2 test = 0.157392678893
```

With just previous years' points and position:

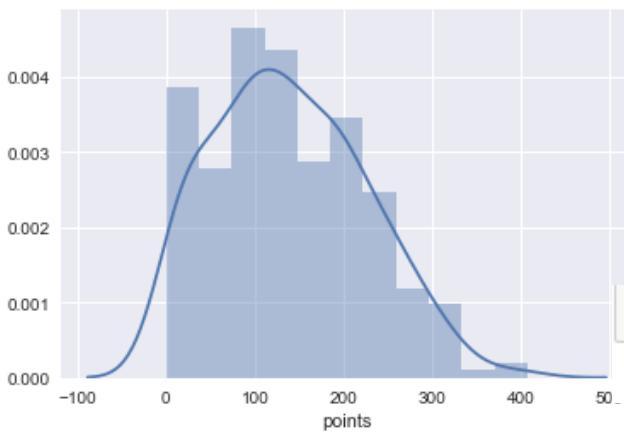
```
adj r2 train = 0.340115683614
adj r2 test = 0.19374638751
```

With ESPN rankings and position:

```
adj r2 train = 0.394117973532
adj r2 test = 0.284009707081
```

Future improvements

1. Include more sites offering predictions (Yahoo, FFT, etc.)
2. Include injury reports
3. Use a different model. Why? My model isn't actually valid because the y-values (player points) aren't normally distributed, even after taking the logarithm. D'Agostino's test confirms a lack of normality:




```
print(normaltest(df['points']))
```

```
NormaltestResult(statistic=9.3182465096389606, pvalue=0.0094747657164112858)
```

Tools Used

1. BeautifulSoup for four of the websites

2. pd.read_html for the other three websites

3. Sk-learn for all statistical analysis

4.. Fuzzy wuzzy for text recognition:

```
print(fuzz.partial_ratio('Robert Griffin', 'Robert Griffin III'))
print(fuzz.partial_ratio('Seahawks', 'Seattle Seahawks'))
print(fuzz.partial_ratio('Terrelle Pryor', 'Terrelle Pryor Sr'))
```

```
100
100
100
```