Forecasting Daily Fantasy Gamblers' Picks Based on the News

Introduction

In October of 2015, an employee of the daily fantasy sports (DFS) gambling site DraftKings leveraged his site's user data to win $350,000 on a competitor site, FanDuel, resulting in allegations of insider trading and concern over the potential for further abuses committed by an unregulated industry.

In a zero-sum game, knowledge of opponents' positions (the field's picks) ahead of market (the contest) represents a serious advantage (or serious abuse when applied by the same people making the market). The expected value of pick I, `V(pick_I)` , is a product of the probability `P` of pick I's success and its associated payout `p` :

```
 V(pick_I) = sum_J P(S_J)*p(S_J;w_I).
```

`P(S_J)` is the probability of pick I's score J, `S_J` , and is independent of gamblers' perceptions, whereas payout `p` depends on pick I's score and gamblers' collective valuation of I, where `w_I` represents the "market share" (also "weight" or "ownership") in pick I. `{w_I}` is the quantity of interest here, and models projecting field ownerships ahead of market are the goal of this project.

Apart from the intrinsic neatness of explaining/predicting the decisions a collection of people make given incomplete information and perceived utility, knowledge of athlete market shares can potentially help one accurately project outcomes of entire fantasy contests themselves... Specifically, if one has a "good" estimate of the covariance matrix of athlete fantasy output, one can collect statistics from an ensemble of Monte-Carlo-sampled contests, each one represented in a set of tickets giving rise to the predicted market. A basic application of such a simulation would be ranking the tickets by probability of profitability, etc.

The Long and Short of DFS Mechanics

"Fantasy owners" (the gamblers) submit one or multiple ticket(s) of athletes' names together satisfying a set of constraints (e.g., on the total fantasy salary of the athletes, fantasy positions, etc.) to an online site (a.k.a. the bookie - FanDuel, DraftKings, etc.) which takes a rake and pays out a subset of tickets as a function of ticket score, determined by athletes' accumulated game stats after bets are locked.

Broadly, there are two (very different) contest types: a "tournament" game pays out the top ~20%, with payout odds growing ~exponentially from 1:1 at the ~20th percentile line (a typical first place ticket's return is ~1000:1); a "cash" game pays out the top ~50% fixed 1:1 odds. Naturally, these payouts force fantasy owners to favor higher-variance players (riskier positions) in "tournaments" and lower-variance players (safer positions) in "cash" games. This causes lower-mean/higher-variance distributions of ticket scores in the former and higher-mean/lower-variance distributions in the latter. See this FanDuel tutorial for more (or less).

Project Specifics

What causes gamblers to make the choices they make with the information they have, and how accurately can I predict the field of wagers in a given contest?

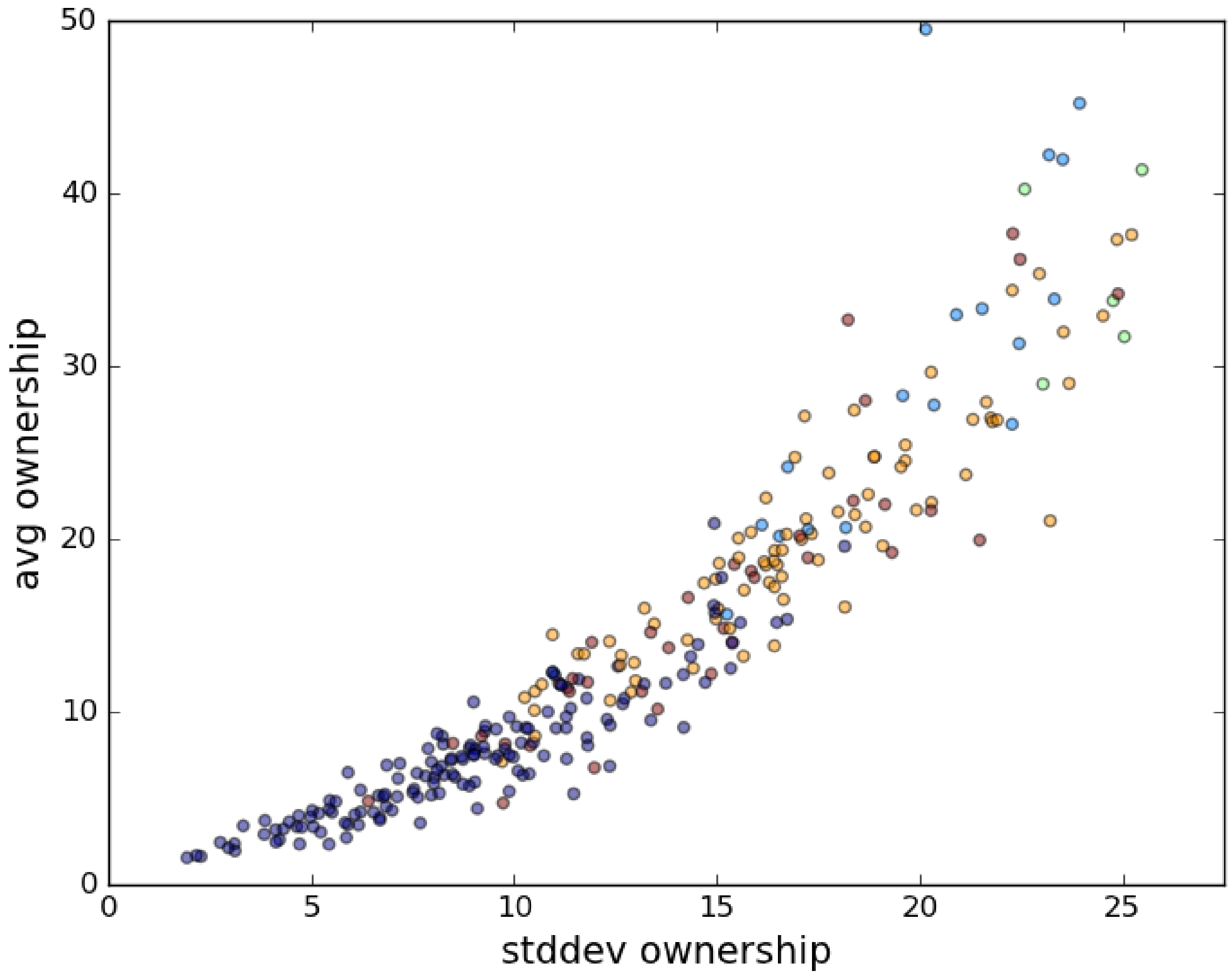
I set out to answer these questions using scraped contest records of field ownerships in past FanDuel NBA contests (thanks to @brainydfs), historic athlete performance data from sportsdatabase.com, the news leading up to the particular contest (e.g., "industry" fantasy output projections from BasketballMonster and FantasyCrunchers, and available injury/roster reports), and elements of the DFS game mechanics presumed to impact gamblers' decisions. (These and others are detailed in the **Models** section below.)

I trained and validated ~400 player-centered models (estimators include ridge, lasso, random forest, and gradient-boosted regressors) on over 700 "tournament" contests from the 2016 season and the first two months of the 2017 season, holding out January 2017 slates for tests.

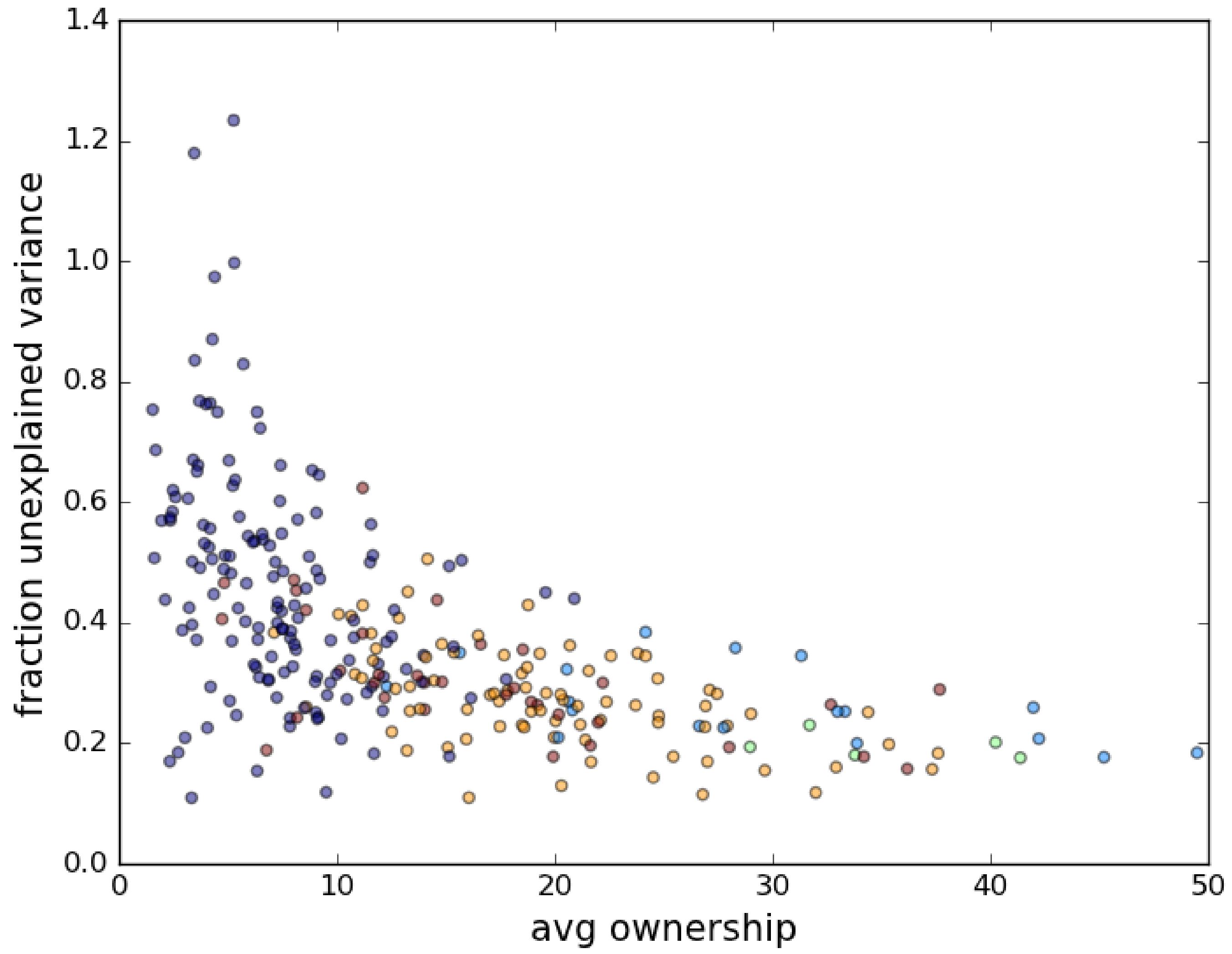
Results

- **training/validating player-centered models**

  Below is a plot of athletes' average ownerships against their standard deviations. Colors correspond to groups coming out of a k-Means decomposition (k=5) of the matrix of player models by respective model coefficients. Clustering of player models (only implicitly included in the model parameters) indicates similar model composition among athletes in close proximity on the mean-standard deviation plane. In other words, it turns out that common factors drive (un)popular athletes' market share. High-mean, high-spread clusters (colored light blue and light green) include Russell Westbrook, James Harden, and LeBron James ownership models, among others you could probably name... Their ownership models depend on, among others, the `slate_size` variable similarly (steeply inversely; see the **Models** section below).



The predictive power of an ownership model increases with mean ownership (and also with the spread around ownership):



This means ownerships of athletes who typically dominate the market share are predicted more accurately. On the other hand, athletes with higher-variance ownerships (specifically, lower mean-to-standard deviation ratio types) are more difficult to predict.

Find your favorite player's stat line here:

name	mean	std	RMSE(train)	RMSE(val)	R2
Russell Westbrook	49.5	20.15	8.14	9.04	0.8
James Harden	45.22	23.94	11.11	10.57	0.81
Kevin Durant	42.23	23.19	9.88	10.9	0.78
LeBron James	41.97	23.53	10.66	11.83	0.75
Kawhi Leonard	41.37	25.48	11.74	10.83	0.82
Stephen Curry	40.24	22.59	9.53	10.43	0.79
Anthony Davis	37.68	22.3	11.15	11.15	0.75
Blake Griffin	37.61	25.23	11.52	10.31	0.83
Damian Lillard	37.33	24.86	7.89	9.48	0.85
Jimmy Butler	36.19	22.47	15.65	9.17	0.83
Giannis Antetokounmpo	35.35	22.95	13.83	10.12	0.81
Draymond Green	34.39	22.28	10.42	10.8	0.77
DeMar DeRozan	34.19	24.89	11.81	11.36	0.79
C.J. McCollum	33.88	23.32	10.01	9.92	0.82
Chris Paul	33.79	24.77	10.56	11.76	0.77
DeMarcus Cousins	33.33	21.54	10.21	10.17	0.78
Kristaps Porzingis	32.98	20.91	9.17	12.11	0.66
Gordon Hayward	32.92	24.52	9.06	9.13	0.86
Klay Thompson	32.69	18.24	9.45	9.69	0.72
Kyle Lowry	31.99	23.55	9.62	8.04	0.88

Carmelo Anthony	31.7	25.05	9.96	13.58	0.71
Paul George	31.32	22.45	9.1	13.68	0.63
John Wall	29.64	20.28	7.89	8.53	0.82
Julius Randle	29.01	23.68	8.55	11.71	0.76
Paul Millsap	28.97	23.04	10.44	10.42	0.8
Rajon Rondo	28.3	19.58	10.05	11.75	0.64
Harrison Barnes	28.01	18.66	7.49	8.36	0.8
Eric Bledsoe	27.92	21.65	9.02	9.66	0.8
Andrew Wiggins	27.76	20.35	8.47	9.14	0.8
Kyrie Irving	27.45	18.39	9.27	9.53	0.73
Kevin Love	27.11	17.16	8.24	9.31	0.71
Isaiah Thomas	27.01	21.75	10.5	8.21	0.86
Danilo Gallinari	26.92	21.31	12.39	9.51	0.8
Rudy Gay	26.89	21.91	11.96	10.83	0.76
Chris Bosh	26.79	21.79	8.56	8.05	0.86
LaMarcus Aldridge	26.64	22.28	10.22	11.15	0.75
Devin Booker	25.45	19.65	7.32	9.69	0.76
Thaddeus Young	24.78	18.89	11.6	9.81	0.73
Victor Oladipo	24.75	18.88	8.74	9.19	0.76
Khris Middleton	24.73	16.92	7.29	9.74	0.67
Kemba Walker	24.52	19.64	10.02	7.39	0.86

Nicolas Batum	24.17	16.75	7.77	11.08	0.56
Will Barton	24.17	19.54	8.07	10.75	0.7
Dwyane Wade	23.81	17.76	11.29	10.57	0.65
Derrick Favors	23.72	21.14	11.44	10.69	0.74
Tim Frazier	22.58	18.73	9.87	12.11	0.58
Serge Ibaka	22.37	16.21	8.03	8.53	0.72
Otto Porter	22.21	18.36	11.26	9.88	0.71
Al-Farouq Aminu	22.11	20.29	9.29	9.59	0.78
Jabari Parker	22.0	19.15	9.0	9.65	0.75
Joel Embiid	21.78	18.35	9.91	13.86	0.43
Avery Bradley	21.67	19.92	8.55	7.97	0.84
Wilson Chandler	21.64	20.28	19.6	9.57	0.78
Jae Crowder	21.57	17.99	13.26	11.12	0.62
Brandon Knight	21.41	18.4	11.47	8.9	0.77
Tobias Harris	21.17	17.2	8.67	7.82	0.79
Chandler Parsons	21.05	23.22	13.89	13.11	0.68
Bradley Beal	20.9	14.94	6.35	9.25	0.62
Hassan Whiteside	20.81	16.11	10.34	9.1	0.68
Dirk Nowitzki	20.69	18.67	10.86	11.33	0.63
Reggie Jackson	20.66	18.18	6.92	9.44	0.73

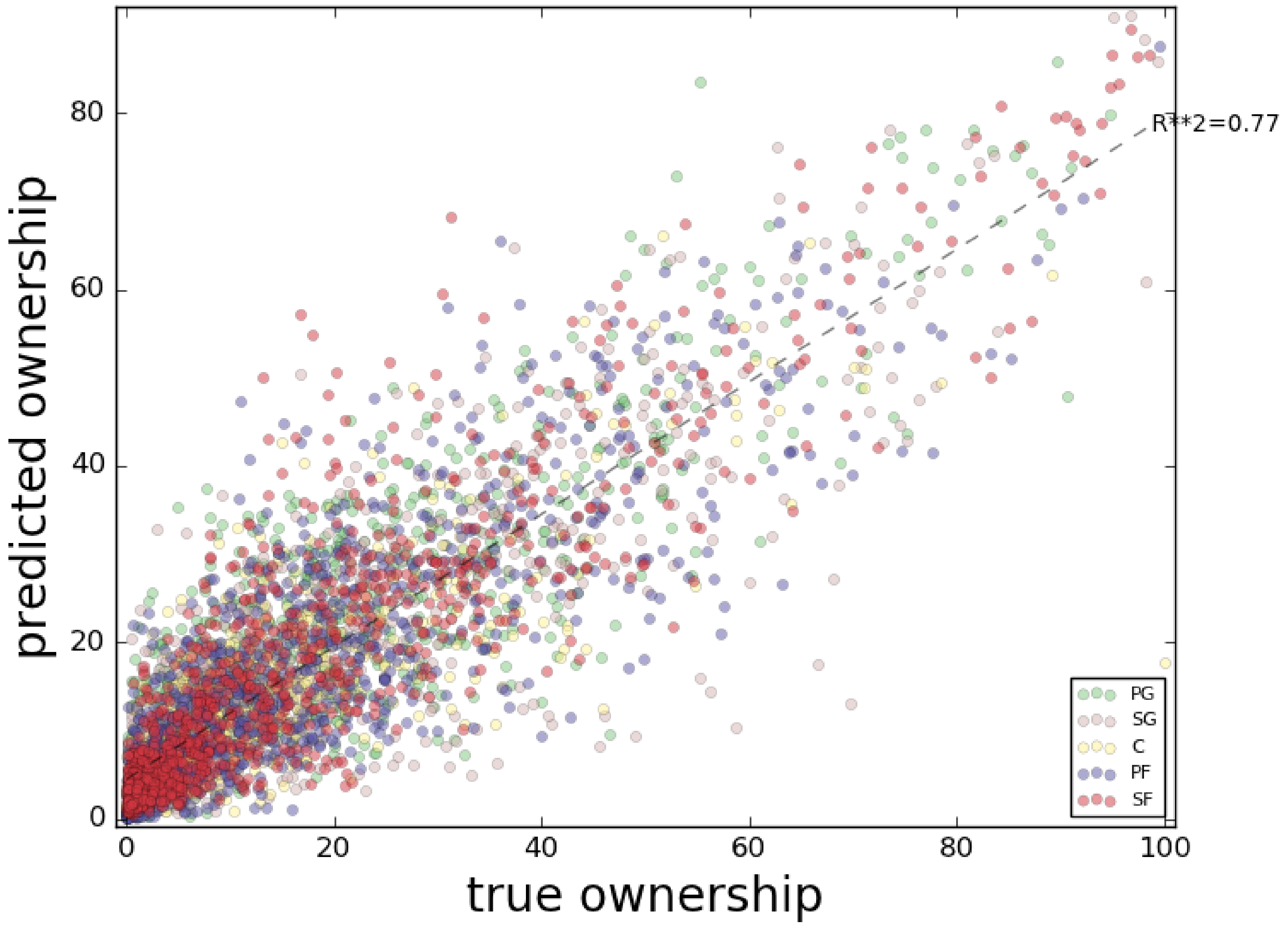
Rodney Hood	20.56	17.25	9.05	10.25	0.65
Karl-Anthony Towns	20.4	15.84	7.26	8.75	0.69
Jared Sullinger	20.31	17.33	10.18	5.99	0.88
Zach LaVine	20.27	16.72	7.92	8.07	0.77
Lou Williams	20.18	17.05	9.95	8.44	0.75
Goran Dragic	20.16	16.54	9.8	8.8	0.72
Andre Drummond	20.04	15.53	8.45	8.0	0.73
Kentavious Caldwell-Pope	19.98	17.09	6.76	8.1	0.78
T.J. Warren	19.93	21.48	9.53	9.89	0.79
Kenneth Faried	19.6	19.09	9.61	10.14	0.72
Evan Fournier	19.58	18.15	11.59	11.87	0.57
Robert Covington	19.36	16.61	8.71	9.33	0.68
Omri Casspi	19.32	16.43	10.34	9.28	0.68
Nerlens Noel	19.22	19.32	8.55	10.56	0.7
J.J. Redick	18.92	15.54	9.13	7.78	0.75
Mike Conley	18.92	17.24	10.37	9.17	0.72
Tyreke Evans	18.78	17.5	13.88	11.75	0.55
Pau Gasol	18.74	16.41	10.56	8.87	0.71
DeAndre Jordan	18.68	16.16	7.28	9.16	0.68
Trevor Ariza	18.59	15.06	7.24	7.06	0.78
Michael Carter-Williams	18.53	15.43	6.9	8.4	0.7

Marcus Morris	18.5	16.49	8.52	7.52	0.79
Jeff Teague	18.48	16.2	9.98	9.07	0.69
Derrick Rose	18.14	15.85	8.33	8.2	0.73
Kent Bazemore	17.82	16.6	7.04	8.96	0.71
Ricky Rubio	17.77	15.92	7.09	7.94	0.75
Monta Ellis	17.77	15.12	8.12	8.76	0.66
Ryan Anderson	17.68	14.97	10.55	8.81	0.65
Emmanuel Mudiay	17.48	16.29	8.28	7.25	0.8
Markieff Morris	17.44	14.68	12.02	8.15	0.69
Gorgui Dieng	17.22	16.42	8.15	9.03	0.7
Zach Randolph	17.04	15.67	7.55	8.36	0.72
Sergio Rodriguez	16.72	12.69	8.12	4.5	0.87
Taj Gibson	16.6	14.29	7.66	8.47	0.65
Shelvin Mack	16.49	16.64	8.85	8.84	0.72
Wesley Matthews	16.15	14.93	5.89	7.57	0.74
Sean Kilpatrick	16.06	18.16	17.78	7.43	0.83
Eric Gordon	15.99	13.21	4.89	6.3	0.77
Nikola Mirotic	15.95	15.03	12.18	6.92	0.79
Myles Turner	15.72	14.95	13.19	10.55	0.5
Jamal Crawford	15.64	15.25	5.2	8.85	0.66

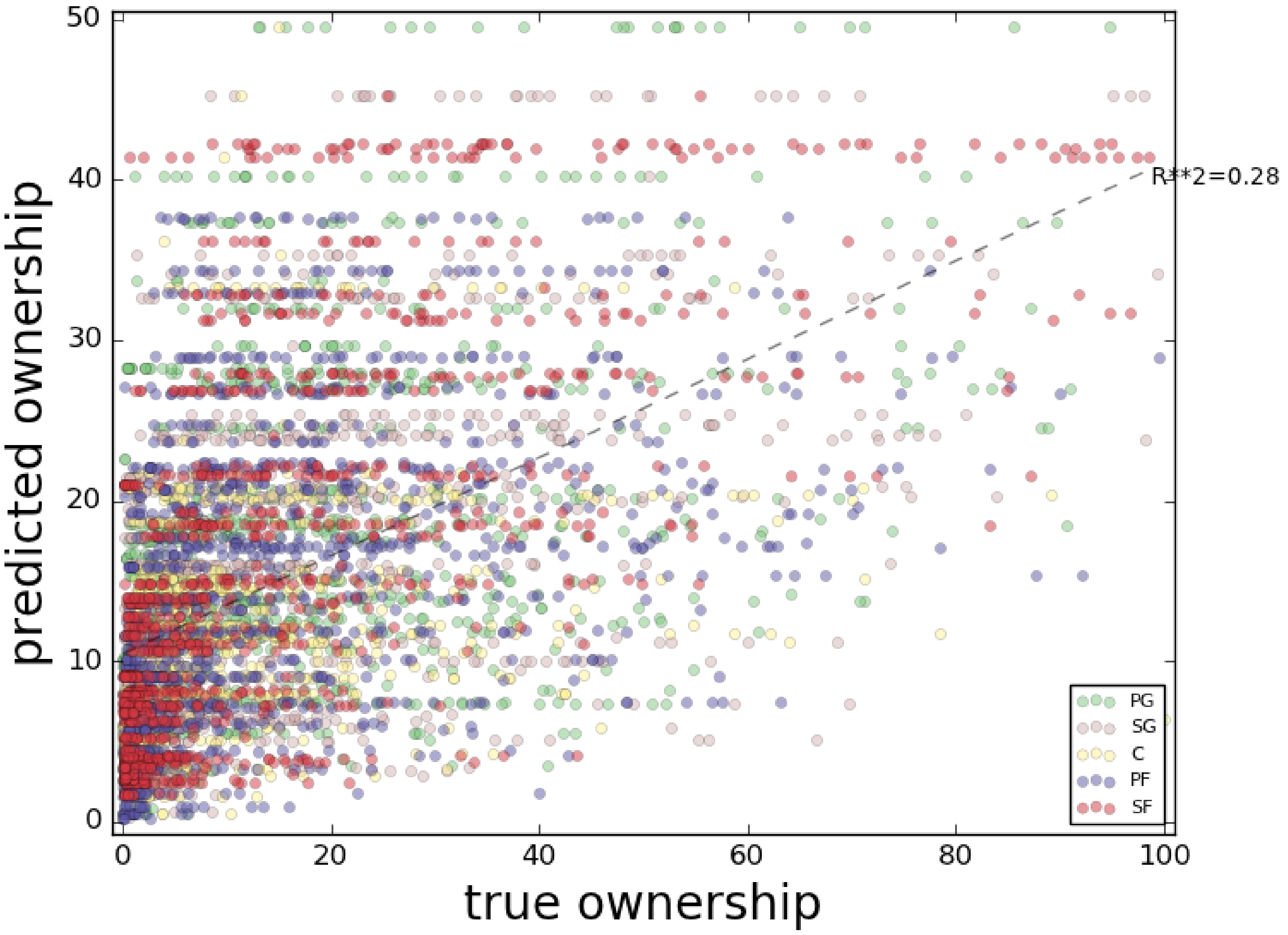
Jordan Clarkson	15.38	14.98	7.11	8.51	0.68
Nikola Jokic	15.35	16.74	7.21	9.24	0.7

- **player models in hold-out**

  I held out contests from January 2017 for testing. Predicted versus true ownerships for each ownership observation are plotted below, with colors corresponding to athlete position, and where perfect predictions would lie on the line `predicted ownership=true ownership` .



Compare these predictions with the null model, which guesses the player's mean ownership each observation:

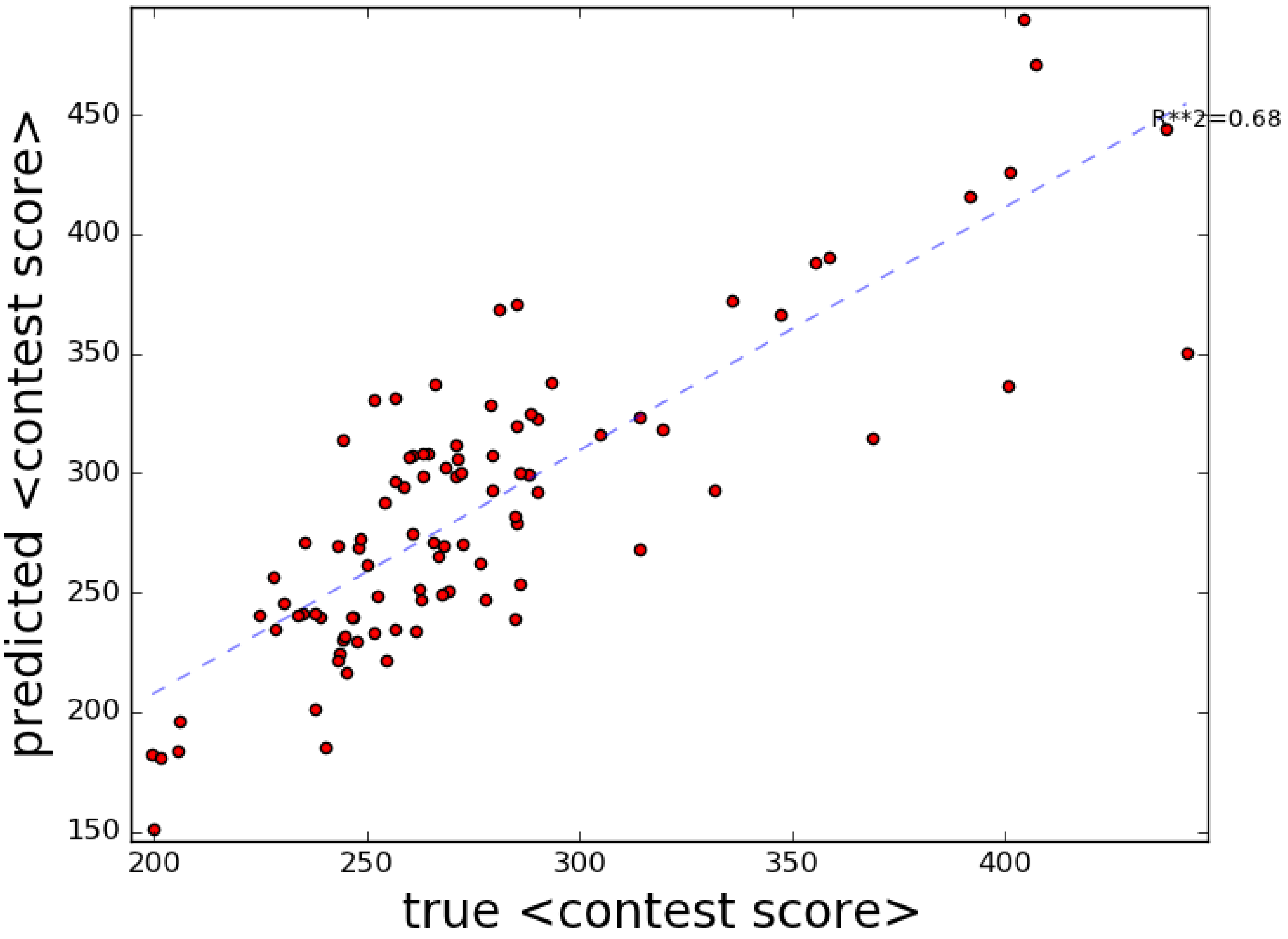


- **bundled predictions of whole-tournament means**

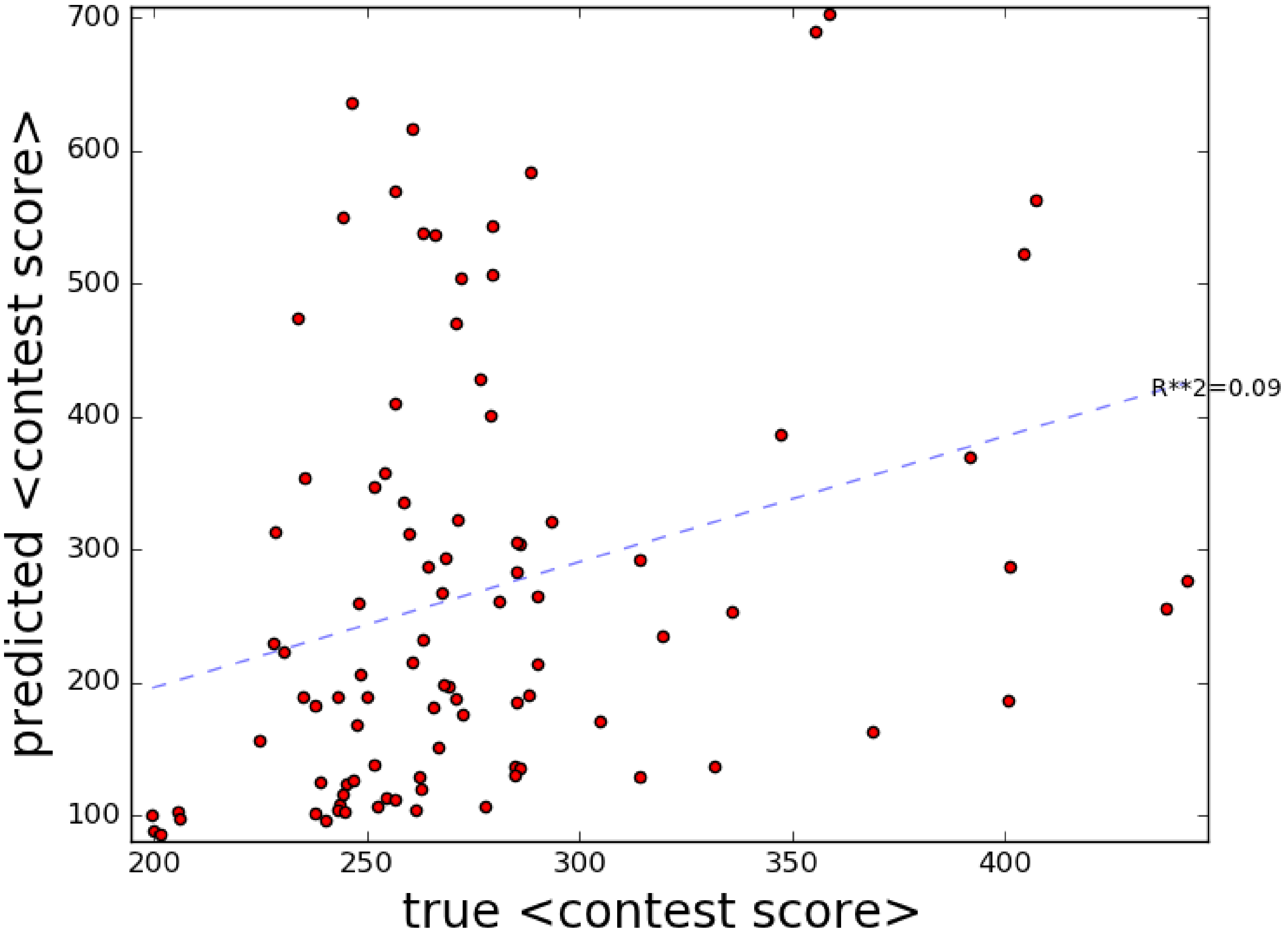
  Player models are doing great, but player models are hardly useful in a vacuum... A useful question to ask is: How well do market share predictions *taken together* predict the market? A contest's average fantasy score `<S>` is an appropriate quantity for this purpose since it couples the `{w}` explicitly,

```
<S> = sum_K w_K*S_K.
```

`S_K` and `w_K` are athlete fantasy scores and ownerships, respectively. In the limit predicted minus true `{w}` go to zero, all that's left to *analytically* determine the exact contest mean fantasy score are the `{S}`, which depend on NBA players alone. Below are predicted versus true contest means for unseen January 2017. Each point is a contest mean computed from the collection of independent player ownerships (actual and predicted) belonging to that contest. Perfect contest mean predictions would lie on the line `predicted < contest score > = true < contest score >`.



Substituting null-model ownerships in the tournament mean equation, predicted means are wildly unrealistic.



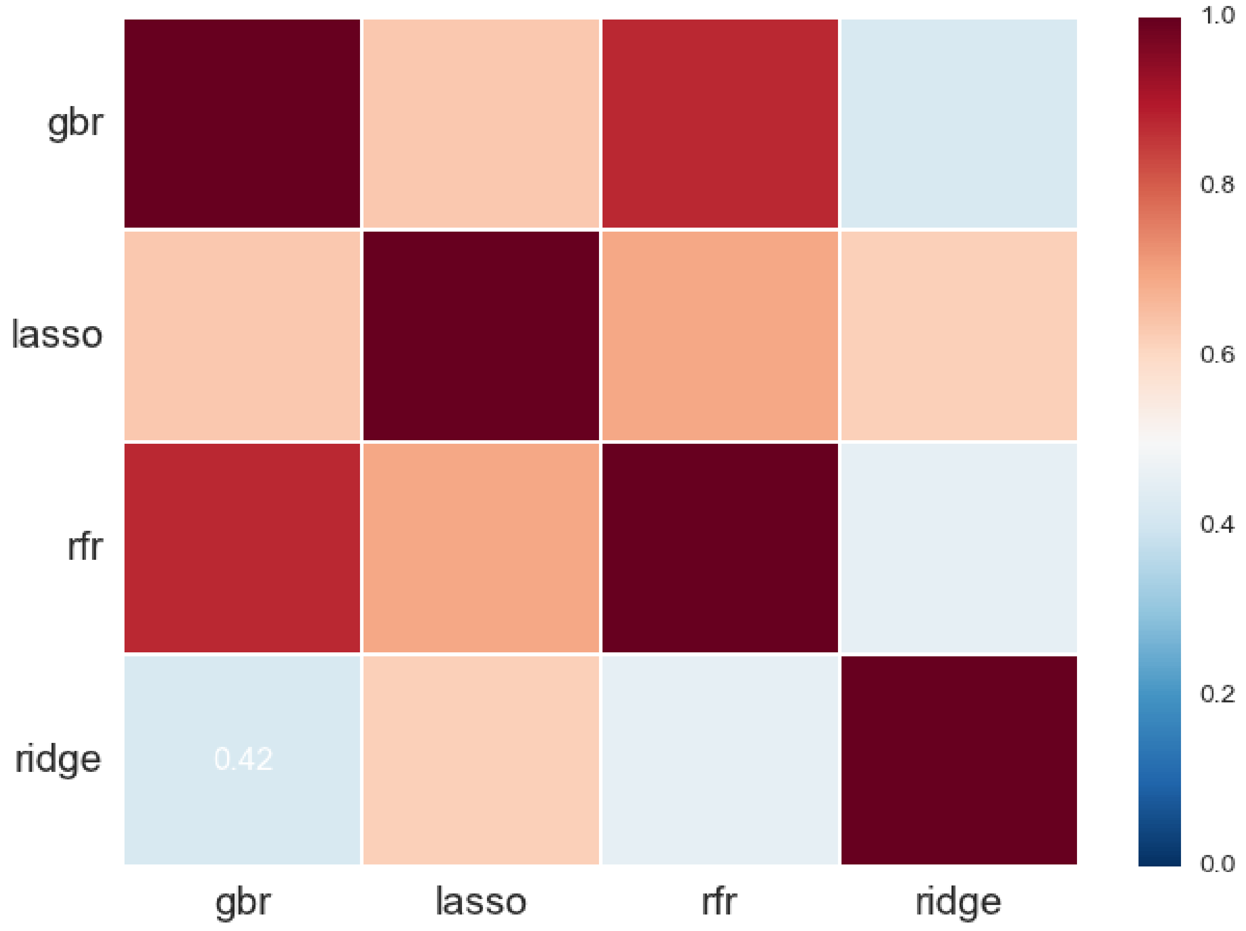
Outlook

In view of their performances in hold-out tests, player ownership models and bundled contest predictions seem to enjoy a great degree of predictive power...

Efforts to improve accuracy might focus on introducing more features, and/or linearly-combining models of player ownerships.

To the first point, given the relatively high impact of "industry" player valuations on ownership models, inclusion of daily projections from more sources, e.g., RotoGrinders, fb-ninja, and any number of others, preferably in order of decreasing volume of their subscription base (after all, the same people buying into industry projections are the same people using these numbers to gamble in contests, and the same people generating the dependent variable modeled here) should increase accuracy.

To the second point, lower-variance ownership predictions are probably obtainable via stacked regression. I trained four estimators: lasso (L1), ridge (L2), random forest, and gradient-boosted regressors. The correlation matrix of their test-set (January hold-out) residuals is plotted below. Surprisingly, lasso and ridge regressors aren't too positively correlated, which isn't the case with the ensemble models. A preliminary investigation of a solution via stacked regression might check errors of an equally-weighted composite model against individual models' errors.



Models

Feature Definitions

Follow the links below to view distributions of regressors for LeBron James' contest observations. (Bear in mind that different variables are important for different athlete models).

1. "industry" valuation
   - `proj_fc` : fantasycrunchers fantasy score projection
   - `proj_mo` : basketballmonster fantasy score projection
   - `v_fc` : fantasycrunchers value (projection /salary)
   - `v_mo` : basketballmonster value (projection /salary)
2. vegas quantities
   - `line` : sportsdatabase matchup line
   - `total` : sportsdatabase matchup total
3. momentum (rolling mean – abbreviated below as "rm" – of previous Y=1-,5-game windows; computed with sportsdatabase queries) –
   - `rm.Y.score` : recent score
   - `rm.Y.salary` : recent salary
   - `rm.Y.value` : recent value
   - `rm.Y.val_exceeds.X` : recent value has exceeded X=4,5,6
   - `rm.Y.opp_total_score` : recent opponent total score
   - `rm.Y.opp_off_score` : recent opponent total offensive score
   - `rm.Y.opp_def_score` : recent opponent total defensive score
   - `rm.Y.team_total_score` : recent team total score
   - `rm.Y.team_def_score` : recent team total defensive score
   - `rm.Y.team_off_score` : recent team total offensive score
4. value over replacement (standard score within X=salary,position class) –
   - `z.X.proj_fc` : z-score of fantasycruncher projections within class X
   - `z.X.proj_mo` : z-score of basketballmonster projections within class X
   - `z.X.v_fc` : z-score of fantasycruncher value within class X

- `z.X.v_mo` : z-score of basketballmonster value within class X

5. game specs/mechanics –

    - `total_entries` : total contest tickets
    - `max_user_frac` : maximum tickets per user / total contest tickets
    - `slate_size` : number of NBA games in slate
    - `log.slate_size` : log(number of NBA games in slate)

6. **fictituous gambler portfolios** (X=worldview,Y=max overlap w/previous solution,Z=number of tickets) –

    This type of feature is arguably the most interesting, and without a doubt among the most predictive. It is constructed as follows: For each contest, initialize a set of "fictitious gamblers", each with a specified "worldview" (tuning bias), some measure of "risk tolerance" (tuning variance), and number of bets. For each fictitious gambler, solve the integer programming problem of constructing a number of unique bets, each maximizing projected fantasy points (enumerated by worldview) subject to feasibility constraints (FanDuel's salary cap, position requirements, etc.) and risk tolerance constraints (maximum number of athelete overlaps with previous integer programming solution in fictitous gambler's portfolio). For each athlete in the slate, construct a feature set of fictitious holdings, each entry of which is the proportion of the fictitious portfolio in that athlete.

    - `gpp.fict.proj_X.Y.Z` : X=(fantasycrunchers,basketballmonster,their average),Y=(2,4,6); Z=25

Feature Importance

To better understand which features are important for ownership models, I computed the percentage of time a feature is in the top 5 over all player models. ('top 5' is defined for ensemble models as the 5 features with the highest gini importance, and for linear models, as the 5 features with the highest absolute-magnitude coefficients). They're tabulated for each estimator below.

rfr feature	%time_in_top_5
gpp.fict.proj_mu.04.025.006	50.9

gpp.fict.proj_mo.04.025.006	41.0
gpp.fict.proj_fc.04.025.006	33.2
gpp.fict.proj_mu.06.025.006	29.1
z.proj_mo	23.0
z.proj_fc	22.3
slate_size	22.0
z.salary	21.5
log.slate_size	21.5
gpp.fict.proj_fc.06.025.006	20.8
gpp.fict.proj_mo.06.025.006	20.5
gpp.fict.proj_mu.08.025.006	18.5
z.sbin.proj_mo	17.0
max_user_frac	16.5
z.sbin.proj_fc	15.2
gpp.fict.proj_mo.08.025.006	13.4
gpp.fict.proj_fc.08.025.006	11.9
rm.01.score	9.1
rm.05.value	8.4

gbr feature	%time_in_top_5
z.salary	52.2

z.proj_fc	32.2
z.sbin.proj_mo	30.4
z.proj_mo	29.6
max_user_frac	27.3
z.v_fc	24.1
total_entries	22.5
z.v_mo	21.3
z.sbin.proj_fc	20.8
z.sbin.v_mo	20.0
z.sbin.v_fc	20.0
slate_size	15.7
log.slate_size	15.2
gpp.fict.proj_mu.04.025.006	13.9
gpp.fict.proj_mo.04.025.006	13.7
gpp.fict.proj_fc.04.025.006	13.4
rm.01.value	9.4
rm.01.score	8.9
rm.05.value	7.8

lasso feature	%time_in_top_5
log.slate_size	55.4

z.salary	43.5
rm.05.val_exceeds.06	22.8
rm.05.val_exceeds.05	20.5
z.proj_mo	18.7
status	18.2
rm.01.val_exceeds.06	16.5
rm.05.val_exceeds.04	15.4
rm.01.val_exceeds.05	15.4
rm.05.salary	14.9
z.proj_fc	13.7
v_mo	13.2
rm.01.score	12.7
v_fc	12.2
rm.01.val_exceeds.04	11.4
proj_mo	10.9
gpp.fict.proj_mo.04.025.006	10.6
rm.01.salary	10.6
gpp.fict.proj_fc.04.025.006	8.6

ridge feature	%time_in_top_5
z.salary	48.6

rm.05.val_exceeds.06	45.3
log.slate_size	44.1
rm.05.val_exceeds.05	41.3
rm.05.val_exceeds.04	33.2
max_user_frac	27.8
rm.05.salary	22.3
rm.01.salary	18.7
status	17.7
z.proj_mo	15.9
rm.01.val_exceeds.06	15.4
rm.01.score	10.6
gpp.fict.proj_mo.04.025.006	9.9
rm.01.val_exceeds.05	9.6
z.proj_fc	9.6
gpp.fict.proj_mo.08.025.006	9.4
gpp.fict.proj_fc.04.025.006	9.1
gpp.fict.proj_mo.06.025.006	8.9
gpp.fict.proj_fc.08.025.006	6.3