Supervised ML - Regression Assignment:

Defining a Relationship for Player Efficiency Rating from John Hollinger's advanced NBA Statistics

Objective

The purpose of this report, and overall study, is to attempt to determine a both a predictive and explanatory relationship between John Hollinger's individual advanced statistics and his Player Efficiency Rating (PER) statistic. It is known that there is a relationship between the 2 variables – however, to this point, the formula for the relationship between PER and the other advanced statistics has not been published. PER is presented as a single advanced statistic, that considers all of a players positive and negative contributions to a game and returns them in a single, weighted measure.

The Data

The data used for the analysis is taken from a Kaggle dataset. It is a singe csv file that summarizes Hollinger's advanced NBA statistics from the 2002 – 03 season to the 2017 – 18 season. The data set is already cleaned, containing no zero, N/A etc data and contains 5404 row entries. Column entries include:

- Rank: that player's PER rank for that given season
- ts%: True Shooting Percentage - what a player's shooting percentage would be if we accounted for free throws and 3-pointers. True Shooting Percentage = Total points / [(FGA + (0.44 x FTA)]
- ast: Assist Ratio - the percentage of a player's possessions that ends in an assist. Assist Ratio = (Assists x 100) divided by [(FGA + (FTA x 0.44) + Assists + Turnovers]
- to: Turnover Ratio - the percentage of a player's possessions that end in a turnover. Turnover Ratio = (Turnover x 100) divided by [(FGA + (FTA x 0.44) + Assists + Turnovers]
- usg Usage Rate - the number of possessions a player uses per 40 minutes. Usage Rate = {[FGA + (FT Att. x 0.44) + (Ast x 0.33) + TO] x 40 x League Pace} divided by (Minutes x Team Pace)
- orr: Offensive rebound rate
- drr: Defensive rebound rate
- rebr: Rebound Rate - the percentage of missed shots that a player rebounds. Rebound Rate = (100 x (Rebounds x Team Minutes)) divided by [Player Minutes x (Team Rebounds + Opponent Rebounds)]
- per: Player Efficiency Rating is the overall rating of a player's per-minute statistical production. The league average is 15.00 every season.
- va: Value Added - the estimated number of points a player adds to a teamâ€™s season total above what a 'replacement player' (for instance, the 12th man on the roster) would produce. Value Added = ([Minutes * (PER - PRL)] / 67). PRL (Position Replacement Level) = 11.5 for power forwards, 11.0 for point guards, 10.6 for centers, 10.5 for shooting guards and small forwards
- ewa: Estimated Wins Added - Value Added divided by 30, giving the estimated number of wins a player adds to a team's season total above what a 'replacement player' would produce.

Outlier data was not removed from the set/each feature, as we did not want to remove important data on shooting % (and other things) that we know are important in determining PER.

Given that the features per, va, and ewa are quite similar in that they describe a players contribution, the va and ewa features were dropped for the analysis. The following were considered for preliminary analysis:

Features

1. ts% True Shooting Percentage

2. ast Assist Ratio

3. to Turnover Ratio

4. usg Usage Rate

5. orr Offensive rebound rate

6. drr Defensive rebound rate

7. rebr Rebound Rate

8. Minutes per game (mpg)

Target

1. PER

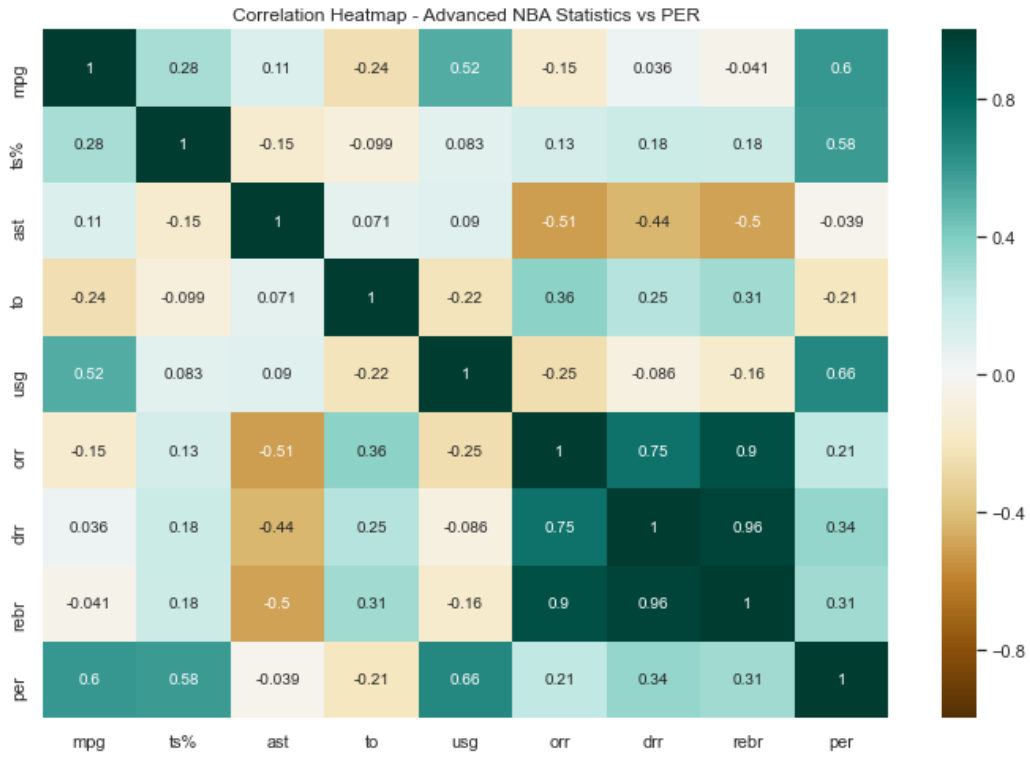
Exploratory Analysis

The first thing done was to get a description of the data. Since all the data was numeric, it is nice o look at the mean, median and different quartiles of the data, to see how it is spread out.

[7]:

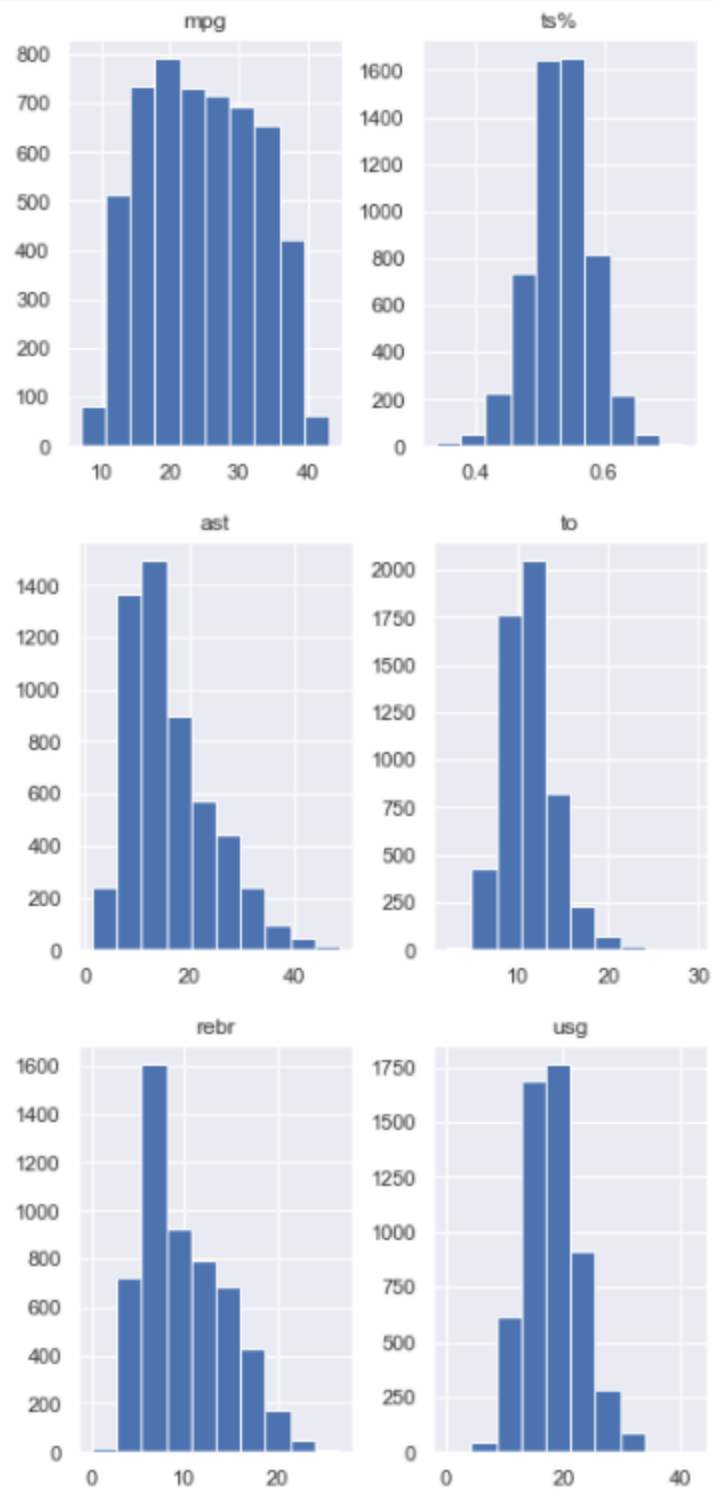
	mpg	ts%	ast	to	usg	orr	drr	rebr	per
count	5404.000000	5404.000000	5404.000000	5404.000000	5404.000000	5404.000000	5404.000000	5404.000000	5404.000000
mean	24.545448	0.531777	15.992746	11.131625	18.066155	5.375093	14.653349	10.013583	14.281956
std	7.954939	0.048009	8.033157	2.880580	4.733323	3.851429	5.817942	4.535166	4.252705
min	7.000000	0.338000	1.200000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	17.900000	0.502000	10.000000	9.200000	14.700000	2.000000	10.000000	6.200000	11.320000
50%	24.400000	0.532000	14.100000	10.900000	17.700000	4.000000	13.600000	8.950000	13.920000
75%	31.200000	0.563000	20.600000	12.700000	21.100000	8.100000	18.600000	13.300000	16.630000
max	43.100000	0.725000	48.700000	29.600000	42.500000	22.000000	38.000000	26.700000	31.760000

Next, a correlation heatmap was built, to show the correlations between features and the correlations between features and the target variable per:



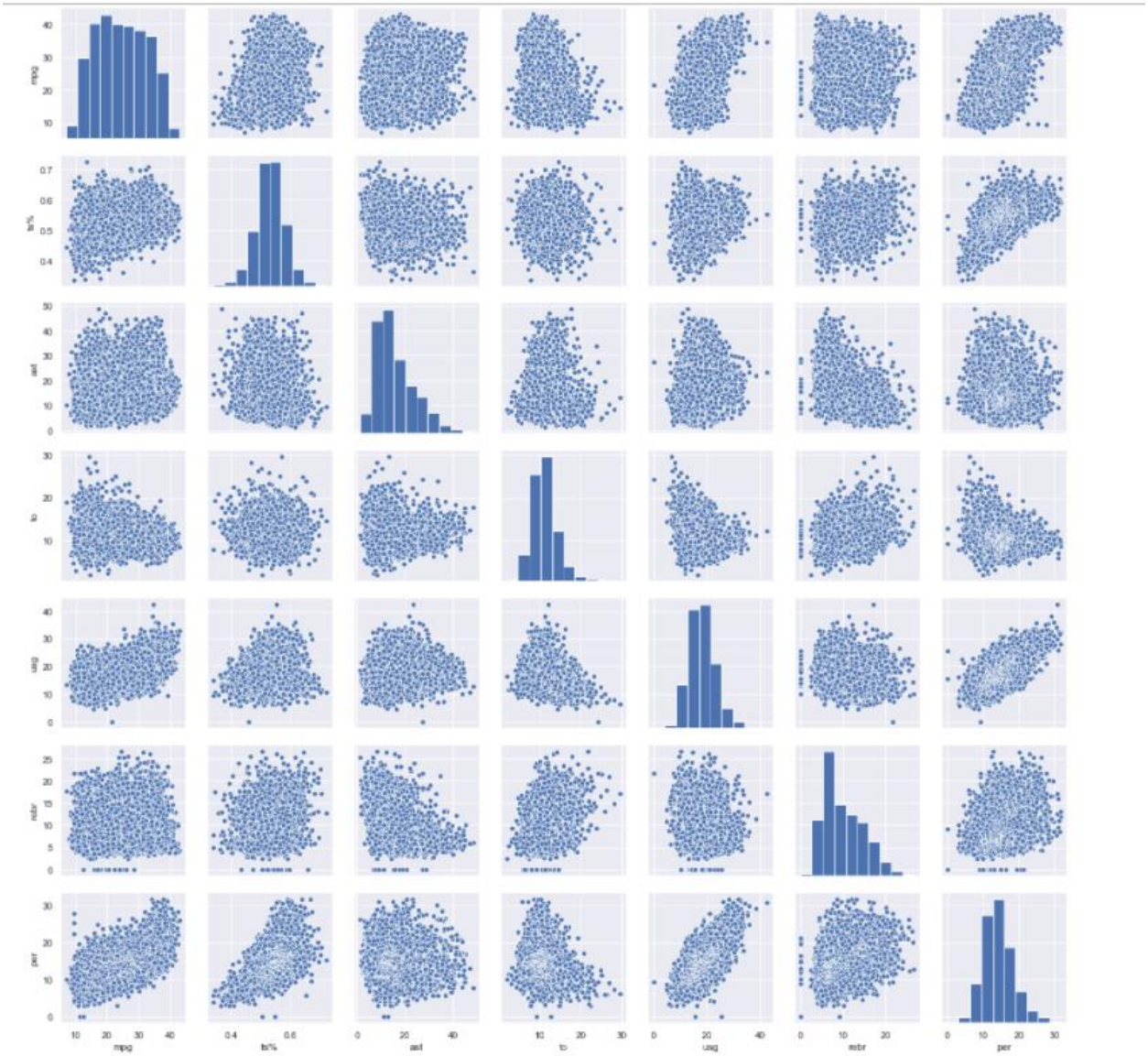
When evaluating the heatmap, it was determined that there were very strong correlations between the features orr, drr and rebr – makes sense since all are measures of rebounding rate. In an attempt to minimize multicollinearity, the decision was made to remove orr and drr, leaving rebr as the only measure of rebounding rate included in analysis. It is understood tat it may cost some explain ability, but it will help in making the regression analysis more ‘clean’.

The finalized features were reasonably distributed and had a lower amount of inter feature correlation, as you can see below:


mpg
ts%
ast
to
rebr
usg

Data Analysis – Regression

See below a seaborn pair plot of the selected features vs per. It is noted that there is some noisy and probably non-linear relationship between the features and per (see the final column of the pair plot



For this reason, 2 separate analyses were carried out

1. Vanilla Linear Regression on the data: all three of Linear, Ridge and Lasso were carried, just to show that Ridge/Lasso do very little in the way of performance when there is little-no complexity to punish
2. Linear Regression on Polynomial Feature (order = 3): again, Linear, Ridge and Lasso completed

The dataset was initially split into a 70/30 train/test split (3782 training records, 1622 testing records) and scaled using the sklearn StandardScaler object. Both the ridge and Lasso models were iterated over a set of 10 equally spaced alpha parameters from 1x 10e-9 to 1.

The results of the regressions, showed (to the 4th decimal point) no difference in R-Squared or MSE for either of the models:

```
Linear Regression Results
R2 Score:  0.8662424314565693
MSE:  2.0062382848547453
----------------------
Ridge Regression Results
R2 Score:  0.8662424314564874
MSE:  2.0062382848550158
----------------------
Lasso Regression Results
Alpha:  1e-09
R2 Score:  0.8662424312910039
MSE:  2.0062382853298337
```

This makes sense – that there would be little – no change in performance in regularizing a model with low complexity (6 feature prediction). The coefficients of the regressions told a little different story, as the Lasso Regression did zero out a couple of the terms. In this case, I'm unsure if Lasso regularization is really helping with interpretability, as it is zeroing out the negative contribution to PER (turnover rate)

[23]:

	features	Linear	Ridge	Lasso
0	mpg	0.553614	0.554288	0.441714
1	ts%	1.796195	1.795744	1.136055
2	ast	0.925834	0.924889	0.000000
3	to	-0.748347	-0.747793	-0.000000
4	usg	2.473734	2.472791	1.583357
5	rebr	2.107803	2.106570	0.412945

If sticking with a plain Linear model, it looks as if either linear or ridge regression work excellently, as they offer the same performance as Lasso, with much better and more intuitive explain ability.

Second, another analysis was carried out on a polynomial feature (degree = 3). As can be seen in the pair plot above, there is definitely a combination of noise and non-linearity in the comparison of each feature vs PER. By visual analysis, and recognizing the desire to keep the linear model as simple as possible, the third order polynomial was chosen.

After fitting a 3rd order polynomial transformation to the feature data, the set was again split to a 70/30 train/test split. Again, after splitting the data was standardized using the sklearn StandardScaler object.

Key Findings

The result of the regressions follows. Of note:

- The linear regression model R-Squared increased about 4 points with a reduction of about 25% in Mean Squared Error, showing that the data relationships did indeed have some non-linearities

```
Linear Regression Score:  0.8662424314565693 Linear Regression MSE:  2.0062382848547453
Polynomial Regression Score:  0.9083500191788483 Polynomial Regression MSE:  1.4581850780209096
```

- The Ridge and Lasso regularizations really didn't provide any type of increased performance – again, likely because the model isn't really that complex to begin with

```
Ridge Regression Results
R2 Score:  0.9089007468279433
MSE:  1.449955432252772
-----------------------
```

```
Lasso Regression Results
Alpha:  0.0001
R2 Score:  0.9088989281449705
MSE:  1.4493829120813084
```

- One of the big differences is the amount of regularization that occurred with the 'optimal' Lasso parameters. It seemed to oversimplify the model, without providing any real bias/variance tradeoff

```
Number of Features:  84
Number of Linear Regression Features:  84
Number of Ridge Regression Features:  83
Number of Lasso Regression Features:  4
```

- Based on the coefficient output of the model, it is difficult to justify the additional computational expense of the polynomial features + regularization:
    - Best case, the R-Squared value went from 0.866 – 0.908, meaning the model went from good (linear) to a little better (polynomial), with a lot of extra analysis. My opinion is that blindly defining a relationship something that can explain 86% of any error is quite good.
    - The explain ability of the model is lost, with so many squared and combination features.
    - The error in predicting PER with the linear model is only slightly higher
        - Taking the square root of MSE to get absolute error, the difference is 0.2 or a PER from linear to polynomial w/ regularization

[35]:

	features	Linear	Ridge	Lasso
0	1	-4.324404e-12	0.000000	0.0
1	mpg	-1.217984e+00	0.242847	0.0
2	ts%	1.000527e+00	0.786844	0.0
3	ast	-2.261291e+00	-0.055560	0.0
4	to	-7.192361e+00	-0.544317	-0.0
...	...	...	...	...
79	to rebr^2	5.596993e-01	0.278693	0.0
80	usg^3	-2.217606e-01	-0.179772	0.0
81	usg^2 rebr	6.983887e-01	0.552174	0.0
82	usg rebr^2	-8.245586e-01	-0.224528	0.0
83	rebr^3	-2.185711e+00	-1.706240	0.0

Further Analysis

As described above, due to the relatively clean nature of the data and the relative few number of features, plain linear regression is likely the best 'bang for the buck' analysis – it provides acceptable accuracy, with an absolute error not much different than that of the regularized polynomial models.

However, there is room for improvement. Due to the nonlinearities/noise in certain relationships (most, except for usage rate and per), a more advanced model may do a better job of defining the relationship – at least for prediction sakes. I would not add any additional features, with the exception of possibly adding in the offensive and defensive rebounding rates – however, that would again point towards a ore complex model, likely less sensitive to multicollinearity effects.