Statistical Learning Project

Lina Lee, Gregory Keslin

1. Strategy Description:

First, we ranked every player that played in both 2019 and 2018 seasons using an overall "fitness" metric. Our metric, in this case, was taking the standard deviations from the mean of each players predicted value in each category in the 2019 season, squaring it, then reapplying the negative so that poor performance is not considered, and then taking the mean across all the categories. We squared each standard deviation in order to emphasize players that are especially good in specific categories. The reason being that we only need to win four out of the seven categories. We standardized the output to make this comparison useful since without standardization we would overemphasize certain categories.

After constructing this overall fitness metric, we next used minimax methodology to form a draft strategy. Since we do not know which players the other teams will draft, our strategy will be to form a decision for each round based on an updated list of the players that every team, including ours, have drafted up to this point. This accounts for the fact that have no idea which players the other teams will draft. Thus, our draft strategy is generalizable to any arbitrary round. Given some arbitrary round in the draft, with a predefined set of players that have been drafted by each team thus far, we can, theoretically, calculate all possible permutations of remaining draft outcomes. Based on this set of permutations we can filter them based on which player we are to pick. For example, if we had a choice between player A and player B in this round, choosing player A then leads to some finite number of possible permutations of subsequent draft picks for the next team and then the next team etc. for all the remaining rounds. The same holds true if we were to pick player B. Then, based on this, we can, for player A, determine which permutation, after picking A, leads to the worst probability of us winning the tournament. To evaluate the probability, we simply calculate, once again for all possible permutations, the predicted total points, rebounds etc. our team has and the predicted total points, rebounds, etc. every other team has. Then, since the tournament is in a round robin fashion, we can predict if we would win four or more categories against each of the other teams. Then our probability for that specific permutation is the number of teams we predict we would beat. Since we do not know which players the other teams will pick, we assume they will pick in the most intelligent manner such that they beat us. Thus, we take the minimum of all the probabilities stemming from all the draft permutations stemming from picking player A. This is the min part of minimax. We repeat this process for player B. We then choose whichever player maximizes this minimum probability. This is the max part of minimax.

Also, since we are predicting multiple rounds into the future, for our own team we do know which player we would pick in subsequent rounds. Thus, in the method we actually used, the permutations we considered were not all possible permutations. Instead, they were all possible permutations assuming we used this draft strategy in subsequent rounds. The other teams though could choose any player. Also, since this was coded in R, recursion, the method we used to implement this strategy, is extraordinarily slow. Thus we can not actually simulate to the end of the draft. Instead, we generally only simulate two rounds ahead and pretend the draft was stopped at this point. This greatly helps reduce the speed needed to compute which player we will be drafting, however obviously leads to more inaccurate results. The second thing we did to speed up

the time complexity of this method was, when generating all permutations, to not allow for other teams to select all the remaining players in the draft. Most likely, a team selecting the worst player in the draft during the first round would not be a viable strategy. Thus, we limited the number of players each time could consider during each permutation. We found five allowed for us to simulate two rounds ahead fast enough. These five players are the remaining best five players, by their aforementioned fitness index, that are undrafted when the team is selecting a player in some permutation. This is the reason why we calculated the fitness metric of every player.

Finally, in our implementation, even though we are only simulating two rounds ahead and only considering, every time a team drafts, the top five players to then permute across, our implementation was coded to be generalized and thus we could, just by changing the variable, simulate across as many rounds as possible and with arbitrarily many players considered by each time at each permutation level.

2. Data cleaning description:

Our datasets include Players, Age, GP, W, L and scores for several categories. We created a dataset data17 by calculating average score per player for each variable using datasets wnba2017 except GP, W, L, and Age. GP, W and L were just summed. The dataset data18 was created by calculating average score per player for the seven categories (Points, Rebounds etc.) respectively using wnba2018 dataset. The two datasets exclude players who did not play in both years. The data17 includes input from wnba2017, and data18 only include average scores of the seven categories from wnba2018. The row number of a player in data17 matches with the row number of the player in data18. We used data17 for predictors and data18 (value of 7 categories) for response to fit the models to predict the score for each category in 2019.

3. Modeling: Variable selection

Our variable selection method was based apriori on our previous knowledge of basketball. Thus, the variables we selected changed based on which category (Points, Rebounds etc.) we were predicting. The included variables for each category are as follows:

We used some predictors for the model for each category. For Points, the model included Age, Min, PTS, FGM, FGA, FG%, 3PM, 3PA, 3P%, FTM, FTA, FT% for the predictors. The Rebounds model included 3PM, 3PA, 3P%, FTM, FTA, OREB, DREB, REB as predictors. The Steals model and the Assists model both included Age, GP, W, L, MIN, PTS, FGM, FGA, FG%, 3PM, 3PA, 3P%, FTM, FTA, FT%, REB, DREB, REB, AST, TOV, STL, BLK, PF as predictors. 3points model has MIN, 3PM, 3PA, 3P%, OREB, DREB, REB as predictors. FT% model has MIN, 3PM, 3PA, 3P%, FTM, FTA, FT%, REB, DREB, REB as predictors. The Blocks model has same predictors with the Steals model and the Assists model.

For certain categories like steals and assists where we were not certain which variables to include, we included all of them. For others, such as three pointers made, we had a strategy for this preselection. Continuing to use the three pointers category as an example, we of course included basic variables like the three pointer statistics and the number of minutes played. However, we also included rebounds as predictors too. The logic being that high rebound players tend to be centers or other larger players that generally are not very good at shooting three pointers. This type of consideration (where we used our previous knowledge of basketball), was used for the points, rebounds, three pointers and free throw percent categories.

4. Modeling: Validation Procedure

we fitted models with datasets data17 and data18. To evaluate the models, we used 10 - fold cross-validation. RMSE was used as the metric to evaluate prediction performance. The models we tried for each category are KNN regression, Lasso, Ridge, Elastic Net (Enet) regression, Principal Components regression (PCR),

Partial Least Squares regression, Smoothing Splines model, LOESS model, and Tree methods. The Tree methods include Tree regression, Bagging, Random Forest, and Boosting. After fitting these models, RMSEs of all the types models for each category were compared and the model with the lowest RMSE has been chosen as the final model for each category.

For the knn regression, a sequence from 1 to 50 by 1 was tried for the tuning parameter k. For the lasso and the ridge model, a sequence from 0 to 5 by 0.1 was tried for tuning parameter λ. For the enet model, we tried different 20 default values for the parameter. In the PCR model and the PLSR model, a sequence from 1 to the number of predictors was applied for the number of components. For smoothing spline regression, a sequence from 1 to 15 by 0.25 was tried for df. For the loess regression, a sequence from 0.3 to 0.9 by 0.2 was applied for span and degree was 1. For the tree regression, a sequence from 0.01 to 0.5 by 0.03 was tried for cp. For random forest, a sequence from 1 to the number of predictors was tried for mtry, and a sequence of (30, 50, 100, 200, 250, 500, 1000, 2000) was applied for the number of tree basically. For the boosting model, the sequence of (0.01, 0.1, 1, 5) was applied for shrinkage, the sequence of (30,50,100,150) was tried for the number of trees, and a sequence of (1,2,3,4) was used for interaction.depth, (3,7,10,12) was tried for n.minobsinnode.

5. Modeling: Final Model for each category

For the Points model, boosting model with n.trees = 50, interaction.depth = 1 shrinkage = 0.1, n.minobsinnode = 3 has the lowest RMSE as 0.4665. The boosting model performs best in predicting next year's points. For the Rebounds model, Boosting model with n.trees = 50, interaction.depth = 1, shrinkage = 0.1, n.minobsinnode = 7 has the lowest RMSE of 0.5916. The boosting model performs best in predicting next year's Rebounds score. For the Steals model, the enet regression with $\alpha = 0.8105$, $\lambda = 0.0992$ has the lowest RMSE 0.6534. Therefore, the model performs best in predicting next year's Steals score. For the Assists model, Ridge regression with $\alpha = 1$, $\lambda = 0.1$ has the lowest RMSE as 0.6056. The model performs best in predicting next year's Assists score. For the 3pts model, the Enet regression with $\alpha = 1$, $\lambda = 0.0337$ has the lowest RMSE of 0.5254. The model performs best in predicting next year's 3pts score. For the FT% model, the enet model with $\alpha = 0.1$, $\lambda = 0.0638$ has the RMSE of 0.6133, and the boosting model with n.trees = 30, interaction.depth = 4, shrinkage = 0.1, n.minobsinnode = 10 has the RMSE of 0.6191. Both have a similar value of RMSE. Therefore, both models perform best in predicting next year's FT% score. For the Blocks model, the ridge regression with $\alpha = 1$, $\lambda = 0.1$ has the lowest RMSE of 0.5524. The model performs best in predicting next year's Blocks score.

6. Modeling: Predictions

Calculated prediction for each player was saved as data frame (filename "predictions2019.RData"). In this data the players are ranked using an overall "fitness" metric, which is described above. The data was used for our strategy. If the model of a category has two types of models which have a similar RMSE, average of the predictions calculated from the two types of models was used as final predictions.

Appendix A: Tables for comparisons of the different types of models

*The model with bold mark is the final model for the category.

Table 1 the Points model

Model	Parameter	RMSE
Knn regression	8	0.5315
Lasso regression	$\alpha = 0$ $\lambda = 0$	0.5310
Ridge regression	$\alpha = 1$ and $\lambda = 0$ .1	0.5254
Enet regression	$\alpha = 1$ $\lambda = 0.03077$	0.5190
PCR	ncomp=1	0.5536
PSLR	ncomp=1	0.5513
Spline regression	df=2	0.5152
Loess regression	span=0.9, degree=1	0.5008
Tree regression	cp=0.01	0.5735
Bagging model		0.5300
Random forest	mtry=3 ntree=2000	0.5142
Boosted	**n.trees=50 interaction.depth=1 shrinkage=0.1, n.minobsinnode=3**	**0.4665**

Table 2 The Rebounds Model

Model	Parameter	RMSE
Knn regression	17	0.6308
Lasso regression	$\alpha = 0, \lambda = 0$	0.6182
Ridge regression	$\alpha = 1$ $\lambda = 0$	0.6182
Enet regression	$\alpha = 0.7157$ $\lambda = 0.0181$	0.6134
PCR	ncomp=1	0.6807
PSLR	ncomp=1	0.6333
Spline regression	df=1	0.6190
Loess regression	span=0.9, degree=1	0.6385
Tree regression	cp=0.01	0.6916
Bagging model		0.6845
Random forest	mtry= 2, ntree=100	0.6129
Boosted	**n.trees=50, interaction.depth=1, shrinkage=0.1, n.minobsinnode=7**	**0.5916**

Table 3 The Steals Model

Model	Parameter	RMSE
Knn regression	k = 16	0.7168
Lasso regression	$\alpha = 0$ $\lambda = 0.5$	0.6824
Ridge regression	$\alpha = 1$ lamda=0.1	0.6541
Enet regression	**$\alpha = 0.8105$** **$\lambda = 0.0992$**	**0.6534**
PCR	ncomp=1	0.7145
PSLR	ncomp=1	0.7035
Spline regression	df=1.5	0.7460
Loess regression	span = 0.9, degree = 1	0.7886
Tree regression	cp = 0.04	0.7639
Bagging model		0.6803
Random forest	mtry = 14, ntree=200	0.6710
Boosted	n.trees = 150, interaction.depth = 3, shrinkage = 0.01, n.minobsinnode = 3	0.6712

Table 4 The Assists Model

Model	Parameter	RMSE
Knn regression	k=15	0.7313
Lasso regression	$\alpha=0$, $\lambda=0$	0.6528
Ridge regression	**$\alpha=1$, $\lambda=0.1$**	**0.6056**
Enet regression	**$\alpha=1$, $\lambda=0.0810$**	**0.6068**
PCR	ncomp=1	0.8814
PSLR	ncomp=1	0.7965
Spline regression	df=1	0.7389
Loess regression	span=0.9, degree=1	0.7557
Tree regression	cp=0.1	0.7613
Bagging model		0.7034
Random forest	mtry=19, ntree=10	0.6904
Boosted	n.trees=30, interaction.depth=1, shrinkage=0.1, n.minobsinnode=112	0.6425

Table 5 The 3pts Model

Model	Parameter	RMSE
Knn regression	K=7	0.5800
Lasso regression	$\alpha = 0, \lambda = 0$	0.5406
Ridge regression	$\alpha = 1, \lambda = 0.1$	0.5324
Enet regression	**$\alpha = 1, \lambda = 0.0337$**	**0.5254**
PCR	ncomp=1	0.8257
PSLR	ncomp=1	0.5571
Spline regression	df = 1.5	0.5336
Loess regression	span = 0.9 degree = 1	0.5434
Tree regression	cp = 0.01	0.5858
Bagging model		0.5318
Random forest	mtry= 4, ntree= 10	0.5541
Boosted	n.trees = 100, interaction.depth = 4, shrinkage = 0.1,n.minobsinnode = 12.	0.5303

Table 6 The FT% Model

Model	Parameter	RMSE
Knn regression	k=21	0.6547
Lasso regression	$\alpha=0, \lambda=0$	0.6133
Ridge regression	$\alpha=1, \lambda=0$	0.6321
Enet regression	**$\alpha=0.1, \lambda=0.0638$**	**0.6133**
PCR	ncomp=1	0.6614
PSLR	ncomp=1	0.6498
Spline regression	df = 1.75	0.6341
Loess regression	span=0.9, degree=1	0.6394
Tree regression	cp = 0.07	0.6709
Bagging model		0.6406
Random forest	mtry=2, ntree=1000	0.6352
Boosted	**n.trees = 30, interaction.depth = 4, shrinkage = 0.1, n.minobsinnode = 10**	**0.6191**

Table 7 The Blocks Model

Model	Parameter	RMSE
Knn regression	k=11	0.6985
Lasso regression	α=0, λ=0	0.5930
Ridge regression	**α=1, λ=0.1**	**0.5524**
Enet regression	**α=1, λ=0.0782**	**0.5530**
PCR	ncomp=1	0.8553
PSLR	ncomp=1	0.7909
Spline regression	df=1	0.6448
Loess regression	span=0.9, degree=1	0.6610
Tree regression	cp=0.01	0.5924
Bagging model		0.6335
Random forest	mtry=19, ntree=10	0.6164
Boosted	n.trees = 30, interaction.depth = 2, shrinkage = 0.1, n.minobsinnode = 3	0.6086