CS 4641 Machine Learning: Final Project Report

Predicting the 2020 NBA All-Stars
with Machine Learning

Bryan Ju Yu Lew

1 Introduction

I am a big fan of basketball. I watch a lot of NBA (National Basketball Association) and I'm very passionate about it. Given the enormous amount of stats available for all things related to the NBA, it was the perfect opportunity for me to make use of these publicly available stats, combine it with what I learn in school and hopefully make something cool.

1.1 Target Problem

At the midway point of every NBA regular season, one weekend is dedicated to be the NBA All-Star weekend, where the best players in the NBA come together over the weekend to participate in numerous events and games. This includes a Slam Dunk contest, a 3-point Shooting contest, and more, including the finale event to close out the weekend – the All-Star game.

The NBA All-Star game is basically the greatest basketball pickup game in the world for that year, featuring the 12 best players from each conference (East and West), battling it out with each other over the course of a full basketball game for millions of fans all over the world.

In the weeks leading up to the All-Star game, 12 players will be chosen from each conference to participate. These 12 players are selected based on votes from the players themselves, coaches, the media members, and fans. While the votes are sometimes based on narrative and fan bias, it is largely safe to assume that the best and most deserving players to make the All-Star teams are usually those that perform the best during the first half of the season.

Given that performance of the players can be measured by in-game statistics, we can make use of these numbers and Machine Learning to predict the top players for each conference, and in turn the All-Stars for that season.

2 Data

2.1 Metrics

There is a huge number of stats available for each and every player, for each and every game. However to ensure that we make the most accurate predictions possible, I picked a subset of these stats which I deem to be the most important stats that best represent the overall game and performance for each player:

Individual Stats		**Team Stats**
Games Played	Steals Per Game	Wins
Games Started	Blocks Per Game	Conference Seed
Minutes Per Game	Field Goal Percentage	
Points Per Game	3-Point Field Goal Percentage	
Total Rebounds Per Game	Free Throw Percentage	
Assist Per Game	Usage Rate	

Table 1: Features picked for each datapoint

The individual stats tells us the player's importance and contributions to his team, and the team stats tells us how the team is doing, which also plays a part in determining whether the players makes the All-Star team.

Predicting the NBA All-Stars is a binary classification problem – each player can either be classified as 1 (All-Star), or 0 (not All-Star). However, the models that we will be using cannot be (easily) specified to classify only 12 players from each conference to be an All-Star. Thus, we will use probabilities instead. The models will predict the probability of each player being an All-Star, and these probabilities will be sorted to give us the top 12 players of each conference, which will fill out our All-Star team rosters.

Data was gathered for each active player starting with the 1984-1985 NBA season (greater availability of stats), up till the 2018-2019 NBA season. In total, there are 14,021 data points, with 14 features each. This excludes all players that do not have statistics before the All-Star game for that year, and also the 1998-1999 lockout-shortened NBA season (there was no All-Star game in 1999). All data was collected from *basketball-reference.com*.

3 Algorithms

The algorithms used to train the models were Random Forests with Bagging, Support Vector Machines and Neural Networks. I will be using the `scikit-learn` library for all of the algorithms.

Decision Trees with Boosting I used `sklearn.ensemble.AdaBoostClassifier` for this algorithm, with the `base_estimator` set to `sklearn.tree.DecisionTreeClassifier`. The parameters that I chose to tune are `n_estimators`, which is the maximum number of estimators to use in the boosting, `learning_rate`, which controls how fast the model trains, and `max_depth`, which controls the maximum depth of the tree and hence its expressiveness.

Support Vector Machines I used `sklearn.svm.SVC` for this algorithm, with a radial basis function kernel. The parameters available to tune were `C`, which is the penalty parameter, and `gamma`, which represents the kernel coefficient.

Neural Networks I used `sklearn.neural_network.MLPClassifier` for this algorithm which allows me to specify the number of hidden layers I want for the neural network and how many perceptrons in each hidden layer to use. Other than the width and depth of the network, I chose to also tune the `learning_rate_init`, which controls the learning rate of the network.

4 Tuning Hyperparameters

Tuning the hyperparameters was all done with `sklearn.model_selection.GridSearchCV`. What this function does is that it allows us to specify a set of parameters which we would like to test with, and it performs 3-fold cross validation with all the different combinations of the specified parameters to find the set that gives the best results.

For each algorithm, the performance of each set of hyperparameters is ranked based on the accuracy which the model gives us based on the cross-validation done on the training dataset. In each of the following subsections, I list out the parameters tested and their corresponding cross-validation accuracy score, as well as the average time taken to perform each fold (to speed up the training time, I configured `GridSearchCV` to make use of all my machine's threads so that the cross validation can be performed concurrently).

4.1 Decision Trees with Boosting

For Decision Trees with Boosting, I first tuned the number of estimators and the learning rate for the AdaBoost algorithm:

learning_rate	n_estimators	Score (Accuracy)	Mean Time Taken per Fold (s)
0.1	10	0.96559	0.43985
0.1	100	0.96449	4.53456
0.1	500	0.96693	22.5046
0.1	1000	0.96727	44.45875
0.01	10	0.96475	0.44779
0.01	100	0.96702	4.39065
0.01	500	0.96693	21.30886
0.01	1000	0.96769	42.29836
0.001	10	0.96357	0.42716
0.001	100	0.96441	3.98067
0.001	500	0.96626	21.75783
0.001	1000	0.96719	36.59395

Figure 1: Decision Trees with Boosting Cross Validation Scores (`n_estimators` and `learning_rate`)

Then the best results were taken and the depth of the Decision Tree was tuned:

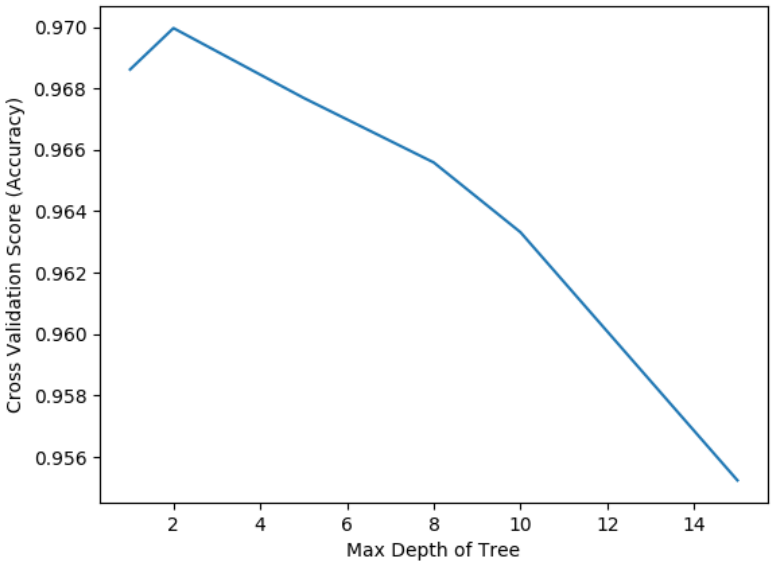


Figure 2: Decision Trees with Boosting Cross Validation Scores (`max_depth`)

These are the final chosen parameters and their test scores:

- `n_estimators`: **1000**
- `learning_rate`: **0.01**
- `max_depth`: **2**
- Test accuracy: **0.977**
- Total time taken (Training, Cross Validation and Testing): **6.4 mins**

4.2 Support Vector Machines

The following table shows the cross validation results for Support Vector Machines:

C	gamma	Score (Accuracy)	Mean Time Taken per Fold (s)
0.1	1	0.93782	28.63413
0.1	0.1	0.93782	21.66477
0.1	0.01	0.96096	3.49271
0.1	0.001	0.96483	2.42227
0.1	0.0001	0.93782	2.74038
1	1	0.93782	36.91046
1	0.1	0.94649	28.78241
1	0.01	0.96777	3.25516
1	0.001	0.96853	2.29222
1	0.0001	0.96584	2.68372
10	1	0.93782	32.40372
10	0.1	0.94691	28.69826
10	0.01	0.96601	3.38098
10	0.001	0.9682	2.4621
10	0.0001	0.96887	2.29722
100	1	0.93782	31.66017
100	0.1	0.94691	28.26454
100	0.01	0.95751	5.11203
100	0.001	0.9687	2.65458
100	0.0001	0.96937	2.65384
1000	1	0.93782	29.96219
1000	0.1	0.94691	26.96024
1000	0.01	0.95675	9.80085
1000	0.001	0.96786	6.01337
1000	0.0001	0.96912	2.52845

Figure 3: Support Vector Machines Cross Validation Scores
(`C` and `gamma`)

These are the final chosen parameters and their test scores:

- `C`: **100**
- `gamma`: **0.0001**
- Test accuracy: **0.976**
- Total time taken (Training, Cross Validation and Testing): **4 mins**

4.3 Neural Networks

For Neural Networks, I first tuned the learning rate of the network:

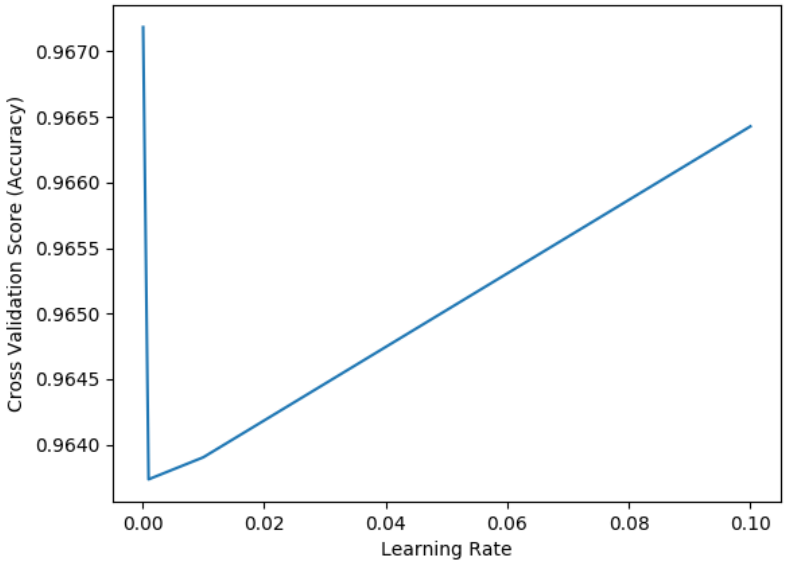


Figure 4: Neural Networks Cross Validation Scores (`learning_rate_init`)

Next, I tuned the hidden layers of the network:

hidden_layer_sizes	Score (Accuracy)	Mean Time Taken per Fold (s)
(50, 50, 50)	0.96584	9.58608
(100, 100, 100)	0.96256	11.98321
(200, 200, 200)	0.96601	33.72318
(300, 300, 300)	0.96517	73.34443
(50, 50, 50, 50)	0.96685	13.64803
(100, 100, 100, 100)	0.96769	25.02051
(200, 200, 200, 200)	0.96508	57.48729
(300, 300, 300, 300)	0.96609	97.41003

Figure 5: Neural Networks Cross Validation Scores (`hidden_layer_sizes`)

These are the final chosen parameters and their test scores:

- `learning_rate_init`: **0.0001**
- `hidden_layer_sizes`: **(100, 100, 100, 100)**
- Test accuracy: **0.972**
- Total time taken (Training, Cross Validation and Testing): **6.2 mins**

5 Comparison

The following table shows the comparison of results between the models of the 3 algorithms used:

	Decision Trees with Boosting	**Support Vector Machine**	**Neural Network**
Test Accuracy	0.977	0.976	0.972
Recall	0.744	0.714	0.767
Precision	0.868	0.872	0.785
F1 Score	0.802	0.785	0.776
Cross Validation Score (95% Confidence Interval)	0.973 (+/- 0.008)	0.972 (+/- 0.006)	0.971 (+/- 0.008)

Table 2: Comparison between the models

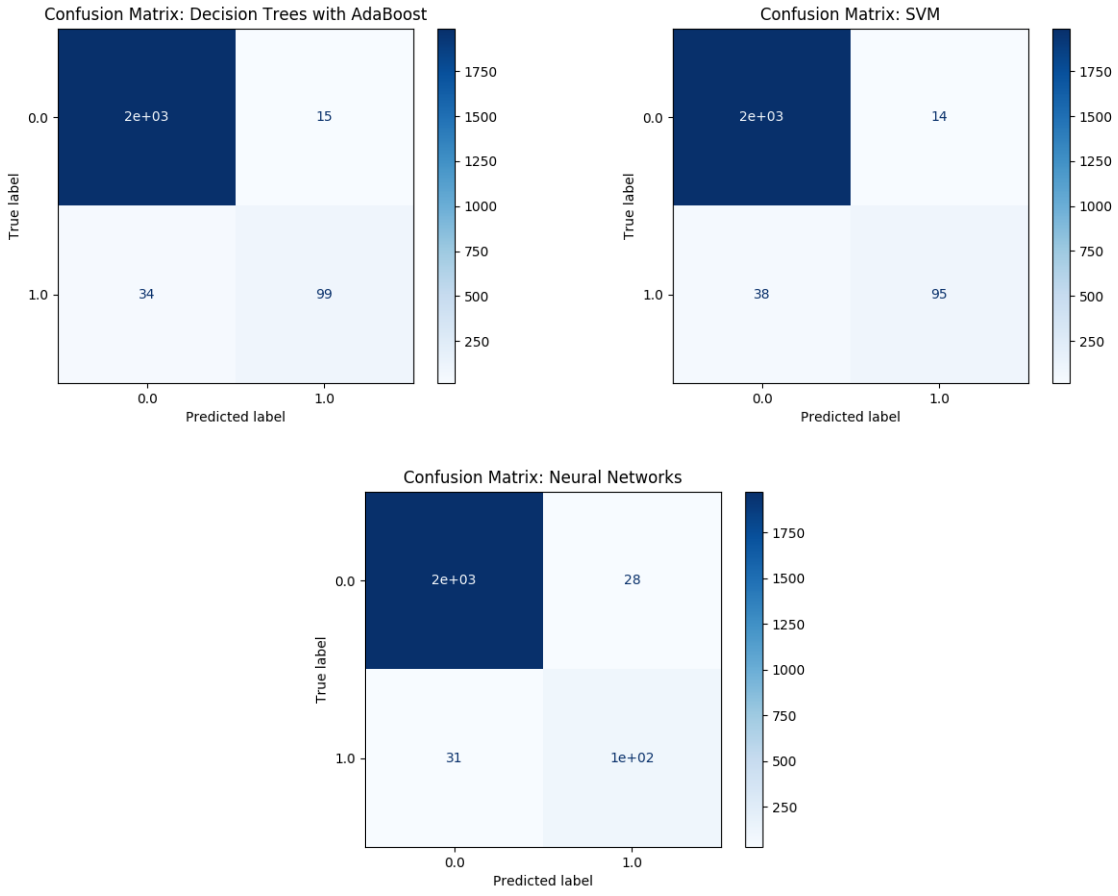
5.1 Model Performance

We can gauge and compare the performance of a model based on a couple of different parameters, such as accuracy of the model on the testing dataset. Across all the three algorithms, the models produced all did fairly well on the test data, achieving more than 97% accuracy across the board, with Decision Trees with Boosting edging out the other two models.

We can also compare the models' F1 scores. The F1 score of a particular model is given by the harmonic mean of the precision and recall, which are the positive predicted value and sensitivity of the models respectively. An F1 score of 1 is considered perfect. Using this metric, Decision Trees with Boosting comes out on top again with the top F1 score of 0.819.

5.2 Confusion Matrices

A confusion matrix allows you to visualize a model's performance based on the recall and precision on the test data. In our confusion matrix, we plot the predicted labels against the true label of each of our data points, and if our model is perfect, we should expect no off diagonal elements, i.e. the closer we are to a diagonal matrix, the better the model's performance.



5.3 Cross Validation Performance

As seen from the table above, the cross validation scores for each of our models are also fairly high and fairly close to the test accuracy of the models. Since none of the models has a cross validation accuracy score that is significantly lower or largely different from the accuracy score on the test dataset, coupled with the small 95% confidence interval, we can say that it is unlikely that the models are overfitting the testing data.

6 Conclusion

6.1 Algorithm of Choice

From the previous section, although all the models performed rather similarly, we can conclude that the best algorithm for our prediction problem is the Decision Trees with Boosting, as it has the highest test accuracy, F1 score and cross validation accuracy score. However, if we do take into consideration other factors such as model training time, we will see that Decision Trees with Boosting takes the longest time to train out of all the 3 (Neural Networks is a close second).

Nevertheless, the algorithm of choice would still be Decision Trees with Boosting, as the accuracy of the model, in addition to its intrinsic benefits (such as ease of implementation, resistance to overfitting), makes it a decent algorithm for our problem domain.

6.2 Predictions

To conclude this report, here are the prediction results with our model of choice, for the 2020 NBA All-Stars:

East Starters	**Probability**	**West Starters**	**Probability**
Giannis Antetokounmpo	0.643071	LeBron James	0.667005
Pascal Siakam	0.605782	Luka Doncic	0.648078
Joel Embiid	0.544134	Anthony Davis	0.610587
Kemba Walker	0.536563	Kawhi Leonard	0.584594
Kyrie Irving	0.535856	Karl-Anthony Towns	0.550807
East Reserves	**Probability**	**West Reserves**	**Probability**
Bradley Beal	0.534351	James Harden	0.619007
Jayson Tatum	0.529155	Devin Booker	0.535276
Trae Young	0.528916	Paul George	0.527977
Jimmy Butler	0.525612	Andrew Wiggins	0.524514
Kyle Lowry	0.501785	Donovan Mitchell	0.520253
Tobias Harris	0.490658	Russell Westbrook	0.519675
Jaylen Brown	0.489267	Montrezl Harrell	0.506492

Table 3: 2020 NBA All-Star Predictions

6.3 What's Next

It is currently still early in the 2019-2020 NBA season, hence I would not say that these predictions will be fully reflective of the actual All-Stars selection in February 2020. The best time to run this prediction again would be somewhere in late January 2020, where we have more stats available and the actual voting begins.

Nevertheless, this predictions should give us a good idea of the All-Stars for this season!

7 Acknowledgements

This project was inspired by *dribbleanalytics.blog*, who has a lot of NBA and data analytics related content.