Multi-Model Basketball Analytics

Patrick Thornton
Author
672 Gunson St
East Lansing, MI
Thorn165@msu.edu

ABSTRACT

Sports predictions is one of the oldest and most popular applications of statistical analysis. In this manuscript I will be detailing the experimental procedure needed in order to create a model that predicts the outcome of a NCAA men's basketball game, starting with data collection and ending with model evaluation. Along the way I will look at many different basketball statistics and discussing how they affect the outcome of a game. After data preprocessing, I will show how selection of appropriate attributes can be used to aid or hinder the construction of a few different classification models. By time this manuscript concludes I will chose, and evaluate, a classification model that, when given a home team and away team, can use statistics, from the previous *k* games, to determine the winner of a matchup, given a home and away team, with 70% accuracy.

Keywords

Preprocessing, Classification, Linear Regression, Decision Tree, Coefficient, Nearest Neighbors, Ada Boosting, Regression, Random Forests, Linear Support Vector Machine, Overfitting, Underfitting, Cross Validation, NCAA College Basketball, Sports Analytics, Machine Learning

1. INTRODUCTION

Across the web you can find research papers on different approaches for predicting modeling of a basketball game. With the recent progress of software, more and more experiments are popping up where researchers are gathering a massive amount of data and utilizing machine learning to process the data and make models. Machine learning, is the process of programming computers to learn from experience [1]. We apply this idea in statistical analysis by supplying a program with a large training set of data and letting it learn, through repetition, how each attribute effects the target attribute.

For this experiment I will be using the Python packages Pandas, Numpy, and sklearn. The data structure I will be using is called a DataFrame, whose documentation is located in the Pandas library. I chose the DataFrame structure for the convenient functions that can be applied to it along with the fact that you can have labels on each column and row if wanted. On top of their versatility, DataFrames are a data structure that can be inputted into sklearn model building functions, which I will be using heavily in this experiment. The aforementioned model building functions all come from sklearn, which has a massive library of machine learning functions and methods. Of the three packages used, Numpy will be the least used in the final project as I will only be using it to create a better version of the default list structure in Python. I will be running my Python code in Anacondas with the Spyder IDE on a windows system.

The paper is laid out as follows. Currently we are in Section 1, which is meant to introduce you to the purpose and method of this paper. Section 2 focuses on introducing concepts and keywords that will be referenced in later section. Next is the Experiment, Section 3. Here is where we get into bulk of this project and talk about the process I went through to create and test my model. Once we have finished the experiment portion of the paper we move to the results section where I talk about the positives and negatives about this experiment. Finally the paper concludes on the conclusion, obviously, where I talk about the take away from this experiment and where to find the code I used. Attached to this paper is the appendix, where you can find any resources I used to or references I make throughout the paper.

2. PRELIMINARIES

One upside to sports prediction experiments is that there is a vast amount of well-documented, in depth statistics. Simple searches online will return any and everything about a sports team; from their teams history to what their star player eats for breakfast every morning. For this experiment I wanted to gather a plethora of game specific statistics from previous games and then, using correlation, eliminate the ones that I did not believe improved our results. To get this information I primarily used a database I found on Kaggle[2], which is normally used in their yearly March Machine Learning Mania competition. With this database I also calculated other attributes I wanted to test, such as the Four Factors[3], which have been shown critical statistics when trying to predict a game. With each game, I would split the data up into home team and away team statistics so that I could focus on how the home team preformed at home and the away team preformed while away.

Once I had the data I started setting up the experiment. First, I wanted to test how each statistics correlated to the points each team got and whether or not the home team won. Correlation is the normalized measure between -1 and +1 that shows how strongly two variables are related. The correlation coefficient between two variables is determined by the function,

$$r(x, y) = \left(\frac{1}{n-1}\right)\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

where x and y are our target variables, ¯and ¯ are the mean values of x and y, and s_x and s_y are the standard deviations of x and y.

After using the correlation coefficient to trim our data down to include only the necessary attributes, I started different kinds of classification analysis. Within analysis there are different types of models, most of which fall under two general types: regression and classification. Regression is a type of analysis where one attempts to explain the variability of a quantitative result based on the variability of different quantitative variables. As mentioned in the prior sentence, the result of regression must be quantitative, and therefore is more useful in an experiment trying to determine an outcome such as the score of a game, which is not the focus of this experiment. In this experiment I will look at classification methods of analysis which uses predictor attributes to predict a nominal-valued attribute. Within classification there are sub-methods of analysis. Of these sub-methods, I tested the Decision Tree, Nearest-Neighbor, Linear Regression, Linear Support Vector Machine, Bagging, Ada Boosting, and Random Forests methods. I wanted to test all these different methods in order to determine which method produces the best results when looking at the outcome of a basketball game. To test these models I first need to determine what split value and *k* value I wanted to use. In this experiment, the split value will refer to the percentage of data that will be in the testing set of the model; i.e. a split value of 0.7 means that 70% of the data will be a part of the set of data I test our model on and 30% of the data will be used to train our model. The *k* value is the number of previous games that our program will pull data from in order to calculate feature vector of statistics. With a chosen split value and *k*, I will select the model I think will perform best to predict a winner. Then I will use hyperparameter tuning and K-fold Cross Validation. Hyperparameter tuning is the process in which you test many different hyperparameters, parameters used to control the model, and choose which hyperparameter settings produce the best result. K-fold Cross Validation is one method used to tune hyperparameters, in which, the training data set is partitioned into K disjoint sets. Once the training set has been divided up, cross validation method goes through each partition, trains a model based on the partition, and tests it against the others. Each time the method gets to a different partition, the hyperparameter are tweaked, this leads to the results from testing each partition to vary. After each partition is tested, the partition with the best accuracy has its version of hyperparameters applied to the rest of the training set which is then used to train on a new model, which will be tested against the test set.

All of these steps are necessary in order to attain a model I can have confidence in and by the end of the experiment I will demonstrate why so much scrutiny is needed.

3. EXPERIMENT

Given the length and nature of analytical experiments, in that if a mistake is made at an earlier part of the experiment it could throw off any later steps, I chose to split the experiment into four steps: data gathering and preprocessing, data testing, model testing, and evaluation.

3.1 Data Gathering and Preprocessing

Before any testing could be done, data had to be gathered. Through research I found many sources of college basketball data. In the end, Kaggle's data seemed to be the most complete and well organized, so it was chosen to be our base database, including every game for every Division 1 team for the past 13 years (76636 games) and 34 attributes for each game. I decided to take the data from Kaggle and calculate some variations of Dean Oliver's Four Factors [3]. According to Dean Oliver, there are four statistics for each team that can be used to determine a team's success. Oliver gave these attributes each a weight and gave the set of these attributes the simple name of "Four Factors". The Four Factors are named as follows: Effective Field Goal Percentage (eFG%), Turnovers (TOV%), Rebounding (ORB% or DRB%), and Free Throws (FT%), then he assigned the weights (40%, 25%, 20%, 15%) respectively. Each of these attributes had a specific formula assigned to them, but since they used seasonal data rather than game data I decided to alter his equations to take game data instead. A full explanation of Oliver's Four Factors can be found online [3]. Out of Oliver's Four Factors, I wanted to test the effectiveness of only the Effective Field Goals, and Free Throws which are calculated as follows,

$$\% = \frac{(+ 1.5 * 3)}{\%} =$$

where FG means 2-point field goals, 3P means 3-point field goals, FGA is total field goal attempts, FT is free throws, and FTA is free throw attempts. With the addition of these two new attributes I had our full list of attributes to test. Now that I had our full set of data, I began refining the data into what I wanted to test in my experiment. Since the purpose of our experiment is to be able predict whether a home team would win based on the averages of the home teams previous k home games and the away teams previous *k* away games, I eliminated any games that were on a neutral court. Next, I had to determine the attributes I wanted to include in our model. Given the 34 attributes in the Kaggle database along with the two new ones that were added, I first eliminated date-related attributes. Another issue that the data from Kaggle posed was that they had organized it into winning teams and losing teams rather than home and away. Using Microsoft Excel, I created a new databases that organized the games based on home and away while keeping the order of the games the same. In this new database I replaced the attribute that said whether it was a home or away game for the winning team with a new attribute that simply had a 1 if the home team won, and a 0 if the away team won. For this experiment, I want to predict the outcome of a game given past games, so I needed to find the previous *k* games for each team and average the attribute of those games. To do this I loaded my data into a Pandas DataFrame. Iterating through the DataFrame, for each game I extracted the home and away team and assigned them variables and made two new DataFrames, one for home and one for away. With our two new DataFrames, I checked the rest of the main DataFrame and added any entries where the home team was the same as the one from the game I wanted to predict, to the home DataFrame until the home DataFrame was of size *k.* I did the

same thing with the away team. I then found the mean of each column in our home and away DataFrame, and added them to a new DataFrame where the game ID was the same as the game I was trying to predict. Thus I had our final data set of feature vectors representing the average of each attribute for the last *k* games.

3.2 Data Testing

Before I started throwing out attributes, I wanted to run a preliminary model test to see the kind of results I would get using our full data set of averages. For this preliminary test I decided to use a decision tree model given its simplicity relative to the other models I will look at in this experiment. I decided to test our model at different values of *k*, so I tested *k* in the set of {3,5,10,15,25,50,100} games. I also wanted to test different max depths, which I explain later, such as {2,3,5,8,10,15,25} and at different split values between 0.1 to 0.9. As one might imagine when model testing with this many attributes, the results showed severe amounts of overfitting regardless of the value of *k* and max depths. Overfitting is when the training set's prediction accuracy is too high and the testing set's predictions accuracy is to low, which shows the training set did not to a good job of accounting for the variations found in the testing set. Reasons for overfitting includes, having too many attributes, the training set is to small, or poor modeling. Since I had not begun model testing yet and had tested a wide range of training set sizes, the next step I took was attribute testing.

To test whether or not an attribute had much of an effect on the accuracy, I tested the model with and without each attribute. In the end, this led to me removing multiple attributes that could be replaced by a different attributes who had a larger effect on the accuracy. The attributes removed at this stage included 2 and 3 point field goals made or attempted, free throws made or attempted, and offensive and defensive rebounds. Free throws Ire show not to be as effective as free throw percentage, field goals were replaced with effective field goal percentage, and offensive and defensive rebounds were replaced with total rebounds.

Even after removing these 16 attributes (8 for home, 8 for away), our model was still suffering from overfitting, and so I choose to try and remove more attributes. Given that the rest of the attributes had not shown a negative result on our accuracy, I decided to look at the correlation coefficient between each attribute and whether or not the home team won. To do this I used the built in correlation function for DataFrames in order to create a correlation matrix. With our correlation matrix made, I looked at the correlation coefficient between each attribute and removed any who had a value within 0.25 of 0. Once these attributes were removed, our model no longer suffered from overfitting, instead, it started to suffer from underfitting. Underfitting occurs when both the training and testing set's accuracy it to low, in other words, to many attributes had been removed. Adding the attributes back in and recompiling the correlation matrix, I decided to also look at the values between every attribute and the points each team got. To decide which attributes I would keep I summed the absolute value of the correlation coefficient between the given attribute, whether the home team won, and the score of the away and home team. If any given attribute had a summed score of less than 0.5, I eliminated it. I also tried elimination boundaries other than 0.5 but no other boundaries showed as large of a positive effect on our results when compared to the 0.5 elimination boundary. As you can see in Figure 1, this led to us keeping only 5 attributes for each team: Points (HPts/APts), Assists (Hast/AAst), Effective Field Goal Percentage (HeFG/AeFG), Total Rebounds (HRbd/ARbd) and Personal Fouls (HPf/APf).

```
#################################### #########
######## Correlation Matrix ######## ## Sum ##
#################################### #########
          Winner      HPts      APts   CorrSum
Winner  1.000000  0.453190 -0.429877  1.883067
HPts    0.453190  1.000000  0.336372  1.789562
HeFG    0.446518  0.651427  0.037404  1.135348
HftP    0.140616  0.213870  0.063252  0.417738
HRbd    0.270371  0.295569 -0.141664  0.707603
HAst    0.352292  0.632302  0.050342  1.034935
HTrn   -0.086136 -0.052112  0.078661  0.216908
HStl    0.175887  0.198251 -0.109281  0.483419
HBlk    0.173714  0.136955 -0.115296  0.425966
HPf    -0.229679  0.134912  0.356604  0.721195
APts   -0.429877  0.336372  1.000000  1.766249
AeFG   -0.451756  0.008331  0.667171  1.127258
AftP   -0.156400  0.017727  0.250641  0.424769
ARbd   -0.247642 -0.111418  0.246377  0.605436
AAst   -0.297725  0.106748  0.566586  0.971058
ATrn    0.157169  0.151126 -0.131986  0.440281
AStl   -0.118633 -0.041524  0.122504  0.282661
ABlk   -0.155849 -0.095572  0.095073  0.346494
APf     0.203835  0.314612  0.202225  0.720672
```

Figure 1

With this final set of 10 attributes, our model began showing a consistent accuracy with few cases of overfitting only on models with a max depths larger than 10. With the data preprocessed and the feature attributes selected, it was time to begin our model testing.

3.3 Model Testing

One question I asked myself when I began this experiment was "What model will work best?" and with how easy the Python's Pandas package made testing multiple models, I decided I would make that question part of the experiment.

Since I am using a classification approach to this experiment I decided only to look at classification models. The models I tested include the Decision Tree mentioned earlier along with a Nearest-Neighbors, Linear Regression, Linear Support Vector Machine, Bagging, Ada Boosting, and Random Forests model. For each model I also wanted to test different hyperparameters along with different split sizes. To test split sizes I simply reran the program with different split values ranging from 0.1 to 0.9 having steps of 0.1. After testing all the split sizes I decided on the split value of 0.7, meaning that 70% of my data would belong to the testing set and 30% would belong to my training set. Not only did this produce the highest accuracy score for most of the models, but it also showed little to no overfitting depending on hyperparameter variation. The hyperparameter variation I

just mentioned refers to the different characteristics you can assign to models to improve, or diminish, their results. For example, in a decision tree, the hyperparameter “max depth” refers to the maximum amount a splits the model is allowed to preform before it is forced to return a model. The decision tree model is not the only classification model with hyperparameters, in fact every model I tested allows hyperparameter variation. To utilize these hyperparameters, I created lists of value I wanted to test, such as the list of max depths was 2 through 10, and then re-built the model using a different element within the list, each time storing the accuracy score in a list of accuracy scores. After finding the best accuracy score for each type of model, I put them in a list to be compared with the rest of the models. Figure 2 is the visual representation of the best accuracy scores each of these models produced given a 0.7 split value, and Figure 3 is the decimal value of each accuracy score.

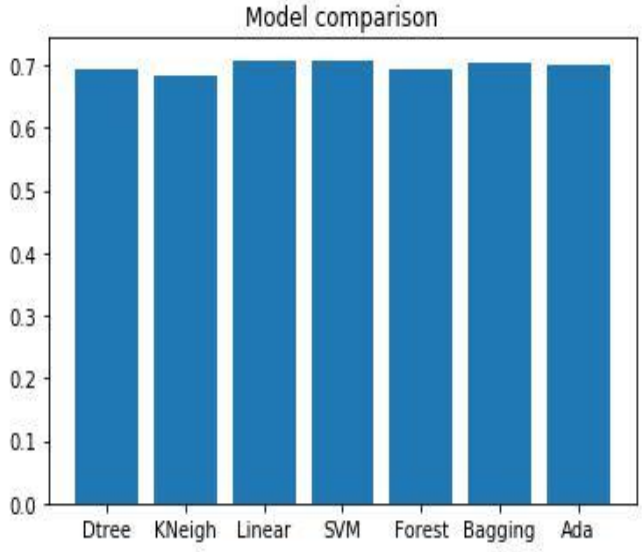

Figure 2

```
############################
### Method ## Accuracy ###
############################
    Dtree :     0.693
    KNeigh :    0.684
    Linear :    0.708
    SVM :       0.708
    Forest :    0.693
    Bagging :   0.704
    Ada :       0.7
```

Figure 3

These figures show that given the split value and attributes, each model achieved a maximum accuracy score of around 0.7, or 70%, with the different hyperparameter settings. Among the different models the multivariate linear regression model preformed with the best results through different random state. Therefore I chose to pursue this model to be the main model I would use in this experiment. Now that I had chosen a model, I wanted test my model to be sure that I had chosen the best one.

3.4 Model Evaluation

As in all scientific fields, the key to result testing is repetition. So the first step I took to evaluate my model was to run it multiple times over to check if the results varied. After determining the results were sound, I decided to move on to the next step, hyperparameter evaluation. Using the cross validation method found in the sklearn package of Python, I tested my models to be sure that I was selecting the best hyperparameter, which I then applied to the rest of the training set in order to achieve the highest accuracy. Not only did I do this on the linear regression model, but also on the SVM model and the Nearest Neighbors model. I chose to do it on the SVM due to how close its accuracy scores Ire to the linear regression models, and I tested the Nearest Neighbors model because I was curious to see the results from a model that performed poorly. Figure 4 shows the results from applying 5-fold cross validation on my linear model. If you are curious like I was, I added the Nearest Neighbors and SVM 5-fold cross validation score in the appendix [4].

```
######################################################
############ Cross Validation of Linear #############
######################################################
Linear Scores:  [ 0.703  0.709  0.71   0.713  0.725]
Accuracy: 0.71 (+/-) 0.01)
```

Figure 4

Given that I was testing multiple different hyperparameter values already, I decided I wanted to visualize its performance at different values. To visualize the different results of my model, I reused the graph I had used to test for overfitting and applied it to my different models. I choose to only show the graph from the linear regression model as it is the model I choose for this experiment, see Figure 5.

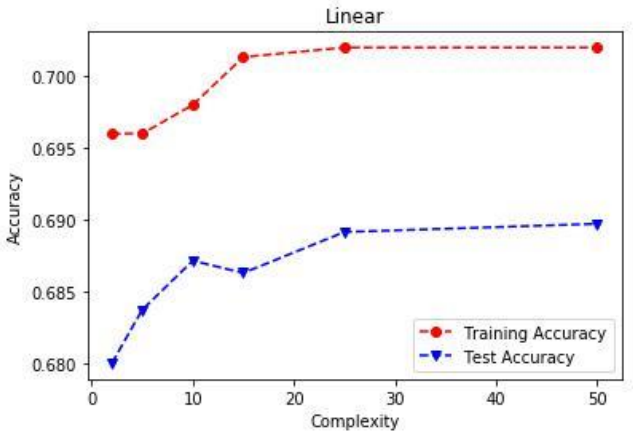

Figure 5

I put the graphs for the other models in the appendix [5].

Through hyperparameter tuning and 5-fold cross validation, the linear regression model held up as not only having the best accuracy score but being a very consistently, well preforming model. Now that I have chosen the best model and proven its validity, I can conclude this experiment on the note that a multivariate linear regression classification model can be used to predict the winner of a basketball with an accuracy of 70%.

4. RESULTS

This experiment clearly shows that there are many methods that can be used to predict the outcome of a basketball better than random selection. Even with that in mind I think a much better model could still be made. During my experiment I had to deal a lot with overfitting and underfitting. However, I doubt that many people would argue that being given more statistics worsens the odds of a true positive result. Thus, I think the next step to improve on this experiment would be to make a model that does a better job accounting for more attributes. I also think that there are a lot of contributors that I did not take into account when starting this experiment. Some of these extra attributes would be easy to implement while others would be near impossible to implement. Of the unaccounted for statistics, the implementable ones include stats such as how many days the team is on the road, if the team is on a winning streak, and luck. Luck would be hard to determine, but once given a value, it would be easy to implement in an analysis model. Some other statistics that have an effect on the game but are much harder to implement would be stats such as whether or not a player is injured, or if there is an event to rally around. By "if there is an event to rally around" I mean there are cases where something happens to a city or to a team and the team begins preforming better than they were expected to. An example of this would be when the Boston Bombing happened the Boston Bruins won the Stanley Cup, or more recently in the NBA playoff when the Spurs head coach's wife died and they went on to beat Golden State Warriors after losing three games in a row to them by double digits.

5. CONCLUSIONS

Even though this experiment is better at picking a winner than random selection, it still underperforms Vegas, and thus there is a lot to improve on. Similarly it is not surprising to anyone that making an analysis model improves the chances of picking a winner, and as such I don't think that should be the main takeaway from this experiment. What I found to be much more useful was to see how the different models preformed given the input data I used. The most important thing to be gained from this experiment should be that different models preform differently and some stats are not as important as others or can be replaced by a better one.

6. APPENDIX

[1] Samuel, Arthur L. “Some Studies in Machine Learning Using the Game of Checkers. I.” *IBM Journal of Research and Development*, 1959, pp. 211–211., doi:10.1007/978-1-4613-8716-9_14.

[2] “March Learning Mania 2017” *Kaggle*, 2017, www.kaggle.com/c/march-machine-learning-mania-2017/data.

[3] Oliver, Dean. *Basketball on Paper: Rules and Tools for Performance Analysis*. Potomac Books, Inc., 2011.

[4] **Cross Validation For Linear Support Vector Machine and K-Nearest Neighbor Mode**

```
################################################
############ Cross Validation of SVC ############
################################################
SVC Scores:  [ 0.704  0.684  0.681  0.685  0.696]
Accuracy: 0.69 (+/-) 0.02)

################################################
###### Cross Validation of K-Neighbors ######
################################################
Scores:  [ 0.677  0.675  0.677  0.688  0.696]
Accuracy: 0.68 (+/-) 0.02)
```

[5] Hyperparameter Tuning Training/Testing Accuracy Graphs