**Analysis of Models for Predicting Individual NBA Games and Season Records**

Edward She, Sean Gao, and Adrian Lin

Professor: Guy Wolf, TA: Nick

**Introduction**

Millions of people around the globe watch NBA games each year. People enjoy watching to see the incredible highlight reel plays, to watch their favorite athletes in action, and to find out the eventual winner. This last motivation behind watching stems from the fact that no game is decided until the game is over. Upsets are not an uncommon occurrence, and even the team with the greatest regular season record in NBA history can lose the championship after being up 3-1 (the 2016 Golden State Warriors). With this in mind, we were curious: how well can NBA games be predicted? To explore this, we used data mining to build and train models that would attempt to predict individual games and season records. In this paper, we will present our findings from these models.

**Methods**

We began by collecting data on historic basketball games. To this end, we used the site basketball-reference.com as our source of basketball game data, as it provides an extremely complete and well-compiled collection of basketball data. To accomplish the data collection, we wrote a script in Python to scrape data from the website, using python package urllib2 and python library BeautifulSoup. From this, we generated csv files of data that contain box scores of all NBA games played in the last six years as well as team statistics for individual seasons, such as total points, total assists, total steals, total blocks, etc. We also made the decision to include only regular season games and left out playoff games, as we felt that regular season games would be a better representation of team skill in the regular season, which is what we are trying to predict. We also accounted for some of the data being from the NBA lockout season, as well as accounting for teams who may have changed their names. From here, we explored several different models for game prediction, which we detail below.

*Elo System*

The first model we explored was an Elo model. This model is based around determining “true team strength” through an updating measure of team strength, the Elo. We decided to use data from 2012 through 2015 as training data, and data from 2016 as the testing data. This very basic Elo model assigned starting Elos to be 1200 for all teams in 2012 and adjusted the Elos by means of the training data. To compute updated Elo scores after a game, we took a “transformed Elo score” corresponding to the team’s current rating, divided by 400, raised to the 10th power. Then, a team’s expected win probability is given by the transformed rating of the team divided by the sum of the transformed ratings of the two teams. From there, we take the true outcomes, subtract the expected outcomes (based on Elo), and multiply by a scale factor K that we default to 32. Then we add this to the original rating to get the updated rating. We trained this on the data from 2012 through 2015 to get our Elos for each team, from which we predicted win-loss records for teams in the 2016 season.

*Logistic Regression*

The next model we explored was a model based on logistic regression. The idea of logistic regression is based in linear regression. In linear regression, we attempt to fit a result vector to a feature matrix based on a linear model, where our output is a vector of beta coefficients. These beta coefficients can then be used to predict new data points. In logistic regression, we use a similar idea. The difference is that we map the result to a logistic curve, which converts the result to a value between 0 and 1. Then, we are able to interpret these values as a probabilistic measure.

In this model, we selected a number of significant features based on the F-test. We used these features to fit a logistic regression model using LogisticRegression from the python library sklearn. Our training data was, similarly to before, the data from 2012 through 2015. With this model, we predicted the winner for each game in the 2016 season. We also predicted win-loss records for all teams in the 2016 season using an expected win formulation. That is, instead of giving a team with >50% win probability in a single game a W on their record and a team with <50% win probability an L, we gave a team with win probability *p* in a given game (the probability directly returned by logistic regression) *p* W’s and *1-p* L’s on their record.

*Support Vector Machines*

After logistic regression, we explored a model based on support vector machines. Using an approach similar to that used in the Introduction to Data Mining Exercises, we performed soft-margin SVM. Our beta for soft-margin SVM was empirically determined. We also explored different kernels before finally settling with the linear kernel, due to factors such as performance and time complexity. The features we used in our SVM were determined in the same way as the features we used in logistic regression. Using our trained model (from data in 2012 to 2015), we predicted the winner for each game in the 2016 season. We also predicted overall win-loss records for all teams in the 2016 season using an expected win formulation that was calculated by using a sigmoid function to map distances to win probabilities.

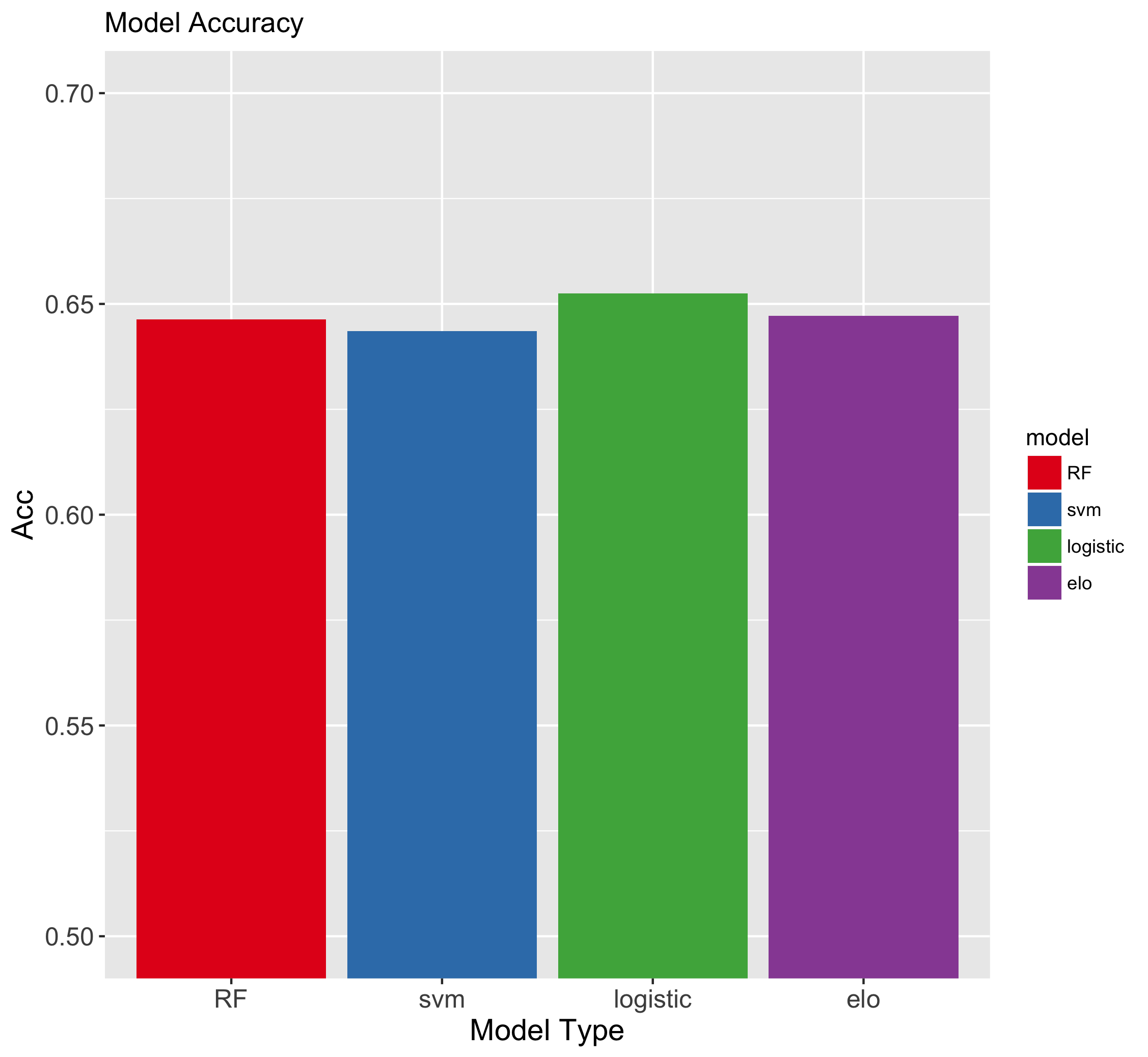
*Random Forests*

Lastly, we explored a model based on random forests. The random forest classification model is based on training a large number of decision trees with different feature selection methods and classifying through majority vote of the decision trees. We performed random forest classification using the RandomForestClassifier package from the python library sklearn. Several parameters were tuned and chosen to maximize accuracy, including the number of trees in the random forest, the minimum number of samples in a leaf node, and the maximum number of features considered for each tree. We trained our model on the data from 2012 to 2015 and predicted the winner for each game in the 2016 season. We also predicted overall win-loss records for all teams in the 2016 season using an expected win formulation that was calculated by computing the proportion of “win” votes across the decision trees.

**Results**

With our overall goal of exploring how well NBA games can be predicted, we devised two main metrics for measuring and comparing the performance of each model. The first metric, accuracy, we devised as a means of estimating how well each model performs in predicting individual games. Since each one of our models was built to predict a single game given historical data, this was the most pure metric we could devise. We measure accuracy simply as the number of games predicted correctly divided by the total number of games predicted. The second metric, root mean square error, we devised as a means of estimating how well each model performs in predicting entire season records. Our models are built to predict a single game given historical data, so a naïve approach is to combine these single game predictions into a season prediction. However, this approach neglects the fact that the confidence level of predictions of different games may vary vastly. That is, we should not consider a prediction favoring one team over another by one percentage point in the same way as we consider a prediction favoring one team over another by forty-nine percentage points. To this end, we converted the confidence level of these individual predictions into expected season records, which we believed would be more accurate. Looking at overall season record projections has a dual benefit in that it is both less noisy and similarly as interesting as individual game winners. With our overall season record projections, we calculated the RMSE of each projected record to the actual record. Each model’s performance under the two metrics is detailed below.

*Accuracy*



**Figure 1. Model Accuracy.** In this figure, we see the accuracy of each of our four models. All of the accuracies are in the 65% range. Logistic regression slightly outperforms the rest.

From Figure 1, we see that the accuracy across the models is very similar. They are all in the ~65% range

*RMSE*

**Discussion**

**Conclusion**