Edward She, Sean Gao, Adrian Lin

Professor Wolf

CPSC 445

18 November 2016

Project Progress Report

For our project on predicting winners of basketball games, we began by collecting data on historic basketball games. To this end, we used 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 this, Sean wrote a script in python to scrape data from the website, using python package urllib2 and python library BeautifulSoup. The python package urllib2 navigates to the url specified and reads the page, while BeautifulSoup parses html. From this, we generated three csv files of data, games.csv, per\_game.csv, and advanced.csv. The first file, games.csv contains the box scores of all NBA games played in the last five years. The second file, per\_game.csv contains players’ per-game statistics, including but not limited to: games, games started, minutes played, field goals, field goals attempted, assists, steals, blocks, rebounds, and more. The third file, advanced.csv contains advanced stats for the players, which we will explore in more advanced models. 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. Using these stats, we hoped to build a model to predict basketball games.

From here, Edward started building a preliminary model to determine “true team strength” using an Elo model. 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. We found that this model achieved 64.7% accuracy overall in our trained data. That is, it correctly predicted 796 games out of the 1230 that were played that season. Our prediction algorithm simply compared Elos and predicted the team with the higher Elo to win (a naïve basic approach). In addition, we did not allow our model to update based on true outcomes of the games; all predictions were done assuming no knowledge of the outcomes of previous games in the season. This is the preliminary model that we will now try to improve on.

For more complex models, Adrian has begun looking at, learning about, and experimenting with python packages that we can leverage to build more sophisticated models. In particular, we will likely be using sklearn to implement some of the following: svms, random forest, regressions, and naïve bayes. In addition, we may also use keras to implement neural networks. Although it is unlikely that we will be able to implement all of these, Adrian has begun evaluating which ones of these may be promising in terms of predictive power.