Predicting the Game: Using Deep Learning and Markov Decision Processes to predict English Premier League Outcomes

Kuhan Jeyapragasan, Madhu Karra, Gita Krishna
AA228 / CS238

Abstract

For years, the English Premier League, arguably the world's most competitive top-tier soccer league, has been difficult to predict and anticipate. This league consists of 20 teams that play each other in a round-robin format over the course of 38 weeks throughout the year. All 20 participating teams are extremely qualified, and consist of the some of the most talented players in the world. This project aims to analyze the factors that contribute to predicting a winner in a Premier League game, as well as factors that contribute to the overall strength of an individual player. To that end, this project has three parts:

Part A: Creating a deep learning predictive model to predict a winner in games given players on the team
Part B: Creating a gambling model to optimize for two specifications using Markov Decision Processes - 1) Maximizing final returns
2) Maximizing probability of reaching a threshold final amount
Part C: Running the K2 algorithm a Bayesian Network to represent how various characteristics influence a player's overall skill

This project will be cross-listed between two classes - CS221: Artificial Intelligence: Principles and Techniques, and CS238: Decision Making Under Uncertainty. For clarity, we have divided our project into three parts: Part A, Part B, and Part C. Parts B and C are most relevant to our CS238 project, and we will elaborate on those more extensively, but we will also discuss Part A as it is relevant to Part B. Part A involves creating a predictive model that can predict a winner in a match-up between two English soccer teams. For this, we created a deep learning model that can predict a winner with approximately 0.73 accuracy. Part B involves creating a gambling agent, and we modeled this as a Markov Decision Process. Our general approach involves modeling states as (money, game) pairs, and actions as betting discrete amounts of money. Part C involves creating a Bayesian Network that assesses how various skills of individual players influence each other, and trying to gather inference based on skills about the overall quality of a player.

Prior Work

There has been a lot of work done in sports predictions, especially for important areas such as the Superbowl, World Cup Final, and other championships. Some previous work done include a random-forest tree approach to predict World Cup Final winners by researchers in Germany, as well as as simple feature-based predictive model based on overall characteristics such as average

goals scored and conceded. There is also a startup based in UK that aims to use AI to make bets on soccer matches. As the CEO of this startup mentioned, sports matches are a great field to explore with AI because "They're short duration, repeatable, with fixed rules; if you observe 100,000 games, there are patterns there you can take out." While these studies contain some similar strategies, ours uses different approaches primarily in terms of the data used. It is not common among this literature to use player data; rather, this literature uses historical team data. Additionally, these approaches focus on one form of deep learning, while we try multiple approaches. In terms of betting, there has been a lot of research conducted, but one that we found particularly relevant to our project is the Kelly Betting method, which can determine the optimal size of a series of bets to optimize the logarithm of money procured.

Part A

Summary

We created a predictive model that, given a set of statistics about two teams in a match-up, can correctly predict the winner approximately 72% of the time. We incorporated a random decision forest model to achieve this.

Data Collection

The data we used came from the FIFA18 dataset, which is a list of around 21,600 players and 80 individual statistics about each player. For example, some of these characteristics included height, pace, dribbling, passing skills, defensive marking, physical aggression, shooting skills, and more. We used the intuition that a team is as good as its players, and decided to represent each team as a vector of the average skill of all its players. Part of this involved creating a mapping from each team to the list of players on that team in a particular season. To create this mapping, we wrote a web scraping algorithm that could gather that information from the internet. We incorporated the Beautiful Soup package, which assisted us in parsing through the html forms of webpages. The particular webpage we used was footballsquads.co.uk. We used a combination of html tag finding and some natural language processing techniques such as clause capturing using regexes and elimination of strings using characteristics such as the length of the string and keywords in the string, and captured a list of players on a team for a given season. We ran this algorithm on all seasons from 2011-12 to 2017-18, and created a dataset mapping from a (team, season) tuple to a list of strings representing all the players on that team that season. Then, using the FIFA18 dataset, we were able to gather all the individual statistics about a player, given their name. Using this, we created a mapping from a (team, season) tuple to a vector that was the average of all the individual statistics of the players on that team. Since

rosters change between seasons, this data approach takes this factor into account when assessing the quality of a team.

We then filtered our data to combine similar features as well as eliminating non-numerical or irrelevant features, and reduced the number of features to 41. As a result, we had every team over the last 6 seasons represented as a vector of length 41, where each value was the average skill value over all that team's players for that particular skill. We were unable to go further back than 6 seasons because beyond the 2011-12 season, there were not many players from our dataset who played in those seasons. As a result, our team vectors would have been an average of 1 or 2 players each, which was not representative of the team. Since there were 380 games per season and 6 seasons, our dataset consisted of 2280 values.

We represented our data input as a vector of length 82 where the first 41 values were Team A's skill vector, and the next 41 values were Team B's skill vector, and the output was 1 if Team A won, and -1 if Team B won. We split our data into training, validation, and test data with the following split: 0.6, 0.2, 0.2.

Predictive Models & Results Analysis

In order to build a predictive model, we tried a number of different approaches. For each model approach, our training data was represented in similar ways as described above. We initially implemented a Logistic Regression predictive model which got us a validation accuracy of 0.7057 over 100 iterations, and a test accuracy of 0.7014 over 100 iterations. Our next approach involved incorporating a Support Vector Machine with an error penalty parameter of 0.35, a kernel coefficient of 1, and an independent kernel function term of 0.5. Over 100 iterations, this got us a validation accuracy of 0.6211 and a test accuracy of 0.6205, which was considerably worse than our logistic regression model. Our next approach involved building a feed-forward neural network. For this, we experimented with various optimizers and losses, and eventually settled on using the stochastic gradient descent optimizer with categorical cross entropy loss. We used a sigmoid input activation function, two hidden layers, and a softmax output activation function. We ran our neural network for 100 epochs, and while the loss steadily decreased, our model did not appear to be learning, as can be observed in the following diagrams.

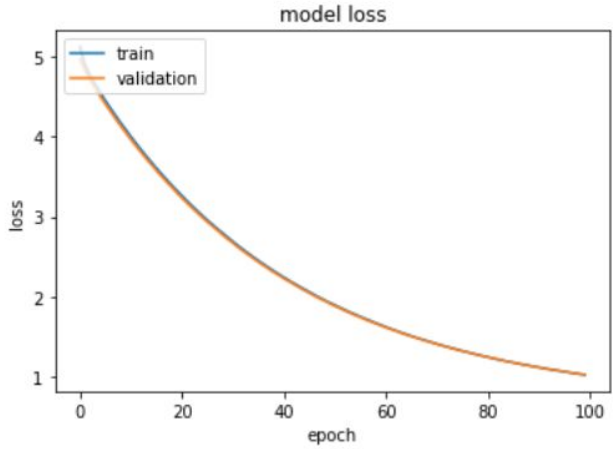

Fig 1. Categorical Cross Entropy loss for training data and validation data

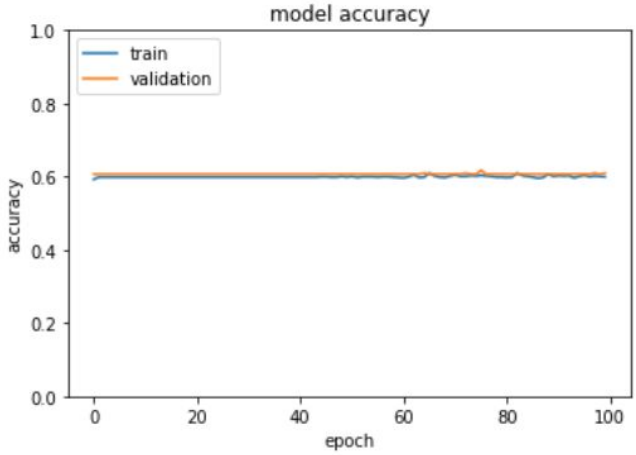

Fig 2. Model accuracy for training data and validation data

We suspect that the relatively small size of our dataset contributed to the fact that the neural network was unable to train well on the data, with a fairly steady training and validation accuracy over 100 epochs. Even though the consistent decline in losses between training and validation data meant our model was not overfitting, this rate of learning was not satisfactory for our purposes. The final model we attempted was a Random Decision Forest Classifier, with a maximum depth of 15, a minimum split of 100, an impurity decrease of 0.001, a minimum leaf size of 30, and bootstrapping. Over 100 iterations, this model performed with 0.7130 validation accuracy, and 0.7478 test accuracy, which was better than all our attempted models so far. Below is a comparison graph of the various accuracies different models provided us.

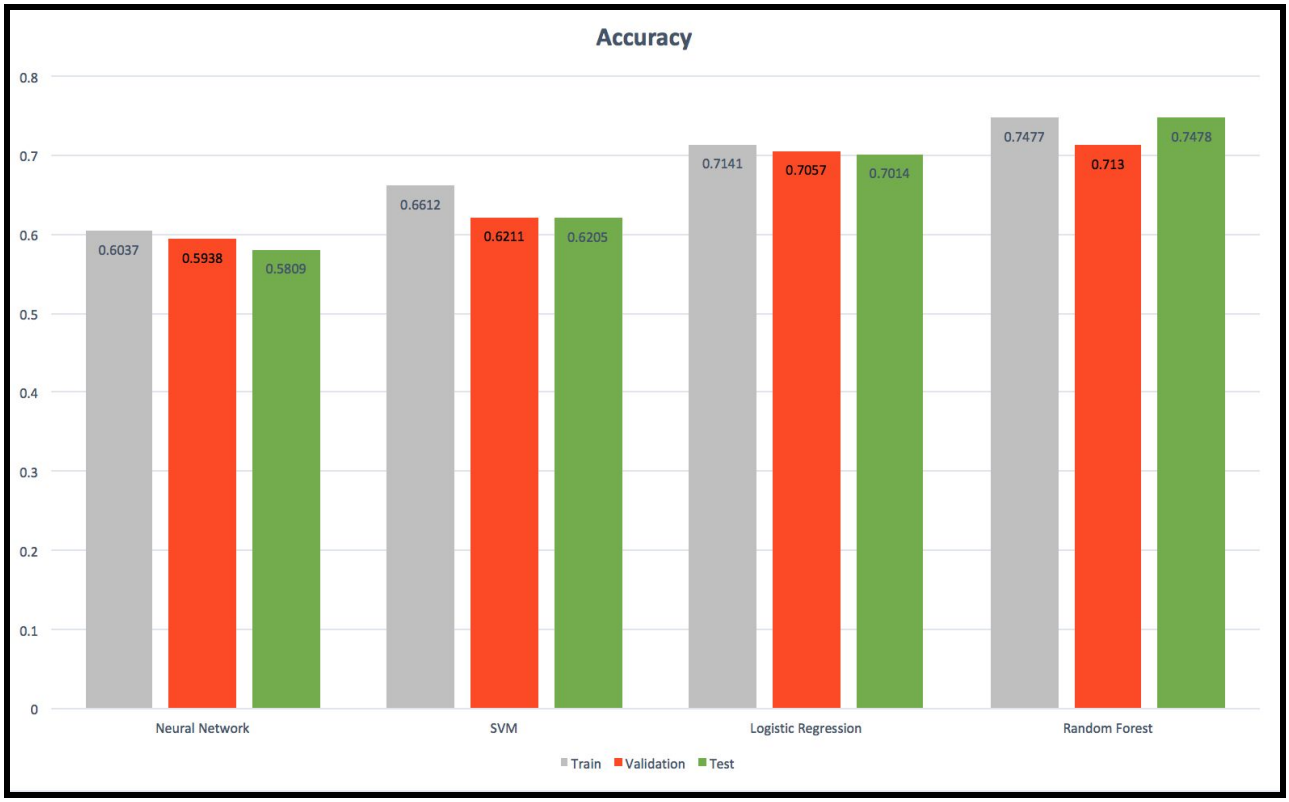

Fig 3: Training, Validation, and Test Accuracies across the 4 attempted models

Our Random Decision Forest classifier outperformed all our other models, followed closely by the Logistic Regression model. The size of our dataset likely played a role in this, as well the size and type of our features. We were pleased to see that none of these models were overfitting, because the accuracies were pretty similar between our train, validation, and test sets.

Part B

Summary

We modeled our gambling problem as a Markov Decision Process. Given that the data we have collected is on soccer teams in the English Premier League, and we wanted to conduct the betting problem in the context of a single knockout tournament, we decided to use the FA Cup to test out our model. The FA Cup is a single-elimination knockout tournament consisting exclusively of soccer teams from the UK.

Our model started with tournaments in the Round of 16 stage, and made a bet on each knockout game until the final, and reported back the final money won. As we found historical

betting odds for the last 15 iterations of the tournament, we compared our predictive model from part A with these historical betting odds to test our betting agent.

Model

For our final MDP, we created a model where a user inputs a starting amount to bet and a goal amount, and our MDP determined a policy that maximizes the likelihood of reaching that goal amount. The state space consists of the game number and the amount of money the agent has (there are fifteen games total, eight in the round of 16, then 4 in the quarterfinals, 2 semifinals, and 1 final). The possible actions consist of betting fractions of the starting money value for each game – (0, 1/20, 1/10 of starting income). Transition probabilities for the system were calculated using our predictive model from part A. Due to the knockout nature of the FA cup, the only relevant transition probabilities were the probability of winning and the probability of losing, which we retrieved from Part A. The final reward for reaching a state with a final amount of money greater than or equal to the goal amount was 100 and not reaching the goal amount resulted in a reward of 0. We then ran value iteration on the state space using a fairly strong discount value of 0.5 (since there are only a finite number of steps available to reach the desired end state, but the specific value will be determined via simulation) to determine the best policy for the MDP. To evaluate our policy, we compared our actions to real FA cup results, as well as comparing our valuation to online betting odds to determine payoff from each match.

While not all teams in the FA Cup were in the Premier league, since our data was for individual players we were often able to create cumulative scores for the teams and compare them to premier league teams. Even still, some teams had to be replaced in our simulation due to insufficient player data. For these replacements, our model would pick the non-replaced team as winner, and not bet any money on the result.

We simulated our model for each of the fifteen iterations of the FA cup, comparing how our predictive scores from Part A fared against online betting odds when fed into our MDP (affecting state transition probabilities) where the better started off with $100 dollars, and each game was able to bet either $0, 5, or $10. The states were defined by (Game number, current money balance). If the model correctly picked the winner, the player would receive twice the betted amount, and otherwise, would lose all money betted. We did not take extra-time/penalties into account and only looked at the final winner. We started off with 16 teams (the round of 16 of the FA cup with some replacements for teams not in the Premier League. Since there were 15 games, the max final score was 250 and minimum was 0, with all multiples of 5 between possibilities as well, so the end state scores were given a reward corresponding to their final score. As well, each game was considered one at a time (not all games in a round were considered in conjunction), so the better would bet on game 1 with no knowledge of future games and make a betting decision, then move on to game 2.

Results

Below is a depiction of how our MDP fared for the 2016-17 FA Cup - starting with $100, our model ended up with $200 out of a possible $250 (guessing each of the $15 games correctly and betting $10 on each):

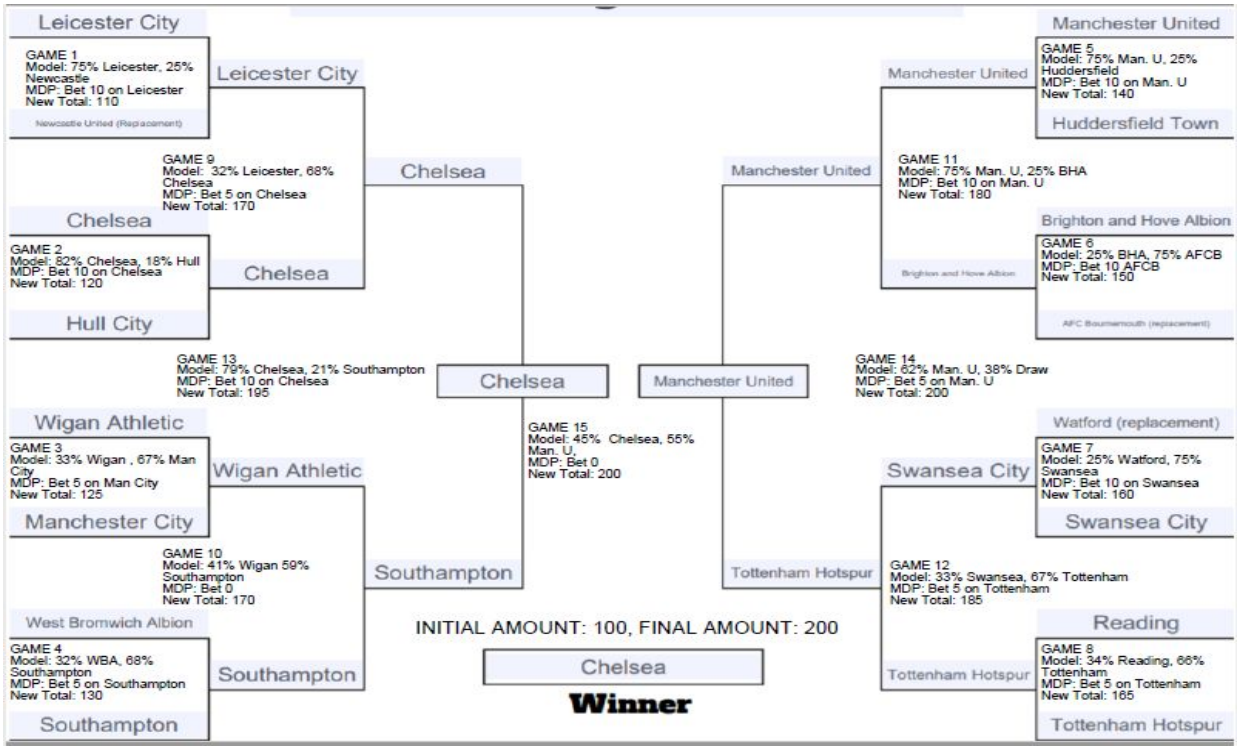

Fig 4: Tournament depiction of the performance of our MDP with states, transition probabilities, actions, and rewards

Overall, our MDP ended up with higher average end values (total reward) in 11 out of 15 (73%) of FA Cups, and was able to reach an end goal of 1.5x initial starting bet more frequently in 9 out of 15 FA Cups.

Analysis

Observing the decisions of our MDP, the action the model decided to take seemed to correspond with the transition probability of victory in the following way:

When the goal was to reach a certain end amount (namely 1.5x the starting amount), on average the policy looked like:

- Bet $10 if Win prob. > 70%
- Bet $5 if Win prob. 60-70%

- Bet $0 if Win prob. < 60%

When the goal was to maximize the final betting amount, on average the policy looked like:

- Bet $10 if Win prob. > 65%
- Bet $5 if Win prob. 55-65%

As can be seen, the policy was more aggressive when trying to maximize its profit rather than just reach a goal value. As expected, when trying to reach a goal value, our program made riskier decisions if earlier bets weren't going according to plan and was conservative when things were going to plan, resulting in slightly worse results than our program just trying to maximize final profits.

Analysis of Prior Work

Comparing our MDP to existing research and literature in the field, the Kelly betting strategy is a formula used to determine the optimal size of a series of bets to optimize the logarithm of wealth. The formula for Kelly betting is:

Fraction of bankroll to wager = (p(b+1)-1)/b, where p is the probability of winning, and b is the net odds received on the wager. Thus with our model of gaining what you bet when predicting successfully, we set b=1, resulting in 2p - 1. Thus, using the Kelly betting strategy, one would want to bet 1/10th of their income when p = 0.55, and 1/5th of their income when p=0.6. This is in concordance with our policy when trying to maximize our betting odds.

Part C: Bayesian Network

Summary

To analyze various features that contribute to the overall strength of a player, we modeled our data as a Bayesian Network to gain further insight into this idea. To do so, we implemented a K2 Search algorithm.

Data

For this part of the project, we used FIFA18 dataset containing information about ~21,600 soccer players. For our first approach, we extracted 41 salient features from the dataset (out of 82 available features) that we used to construct a Bayesian network. The features we used were the following:

['date_of_birth', 'height', 'weight', 'intl_rep', 'pace', 'pace_acceleration', 'pace_sprint_speed', 'dribbling', 'drib_agility', 'drib_balance', 'drib_reactions', 'drib_ball_control', 'drib_dribbling', 'drib_composure', 'shooting', 'shoot_positioning', 'shoot_finishing', 'shoot_shot_power', 'shoot_long_shots', 'shoot_volleys', 'shoot_penalties', 'passing', 'pass_vision', 'pass_crossing', 'pass_free_kick', 'pass_short', 'pass_long', 'pass_curve', 'defending', 'def_interceptions', 'def_heading', 'def_marking', 'def_stand_tackle', 'def_slid_tackle', 'physicality', 'phys_jumping', 'phys_stamina', 'phys_strength', 'phys_aggression', 'weak_foot', 'skill_moves']

In our data, the vector of features above defines one player. The goal was to see if any interesting patterns of influence would arise from this data. Since a Bayesian network encodes conditional dependence assumptions and interactions between these factors, it could reveal insights about how certain qualities of a player lead to success or failure.

Model & Results

We decided to use a K2 search algorithm with a Bayesian score function to find the best Bayesian network. For this algorithm, we started off with an unconnected graph, assuming a Dirichlet prior. We then iterated through an assumed variable ordering, and tried adding every possible parent to one node. We then compute the Bayesian score of this modified network, and check if it receives a score that is greater than the maximum score so far. If it does, we proceed with the modification, else we discard it. Essentially, we find the best parent for each node. We imposed an upper bound of three parents per node. The initial graph received a Bayesian score of -254370.857. After running a several iterations of K2, we had a Bayesian network with a few connected nodes, with a score of -231250.0568. The resulting network shows a relationship between the number of skill moves a player has and the age of a player. Additionally, it appears that there is some relationship between a player's height and their level of physical aggression and physicality.

Fig 5. Resulting Bayesian network

After running this for several iterations, we found that this algorithm was very slow and quite computationally expensive. To remedy this, we grouped together similar features; for example we condensed the features 'dribbling', 'drib_agility', 'drib_balance', 'drib_reactions', 'drib_ball_control', 'drib_dribbling', 'drib_composure' into one feature that represents a player's overall dribbling skill. Once we condensed the features, we were left with sixteen features:

["height", "weight", "pace", dribbling" ,"shooting", "shoot_positioning" ,"passing", "pass_long", "pass_curve", "defending", "def_stand_tackle ", "physicality", "phys_stamina", "phys_strength", "weak_foot", "skill_moves"]

We also made the algorithm faster by caching previous graphs and limiting the number of parents that each node could have to two. We found that the initial Bayesian score for this network was -3368.094, which was significantly lower than the score of the graph with forty-one nodes. When we proceeded with K2, we found that there was no edge that could be added that would increase the Bayesian score; we were left with a completely unconnected graph. We tried many different things, such as modifying the maximum number of parents allowed, changing the number of iterations, and modifying features.

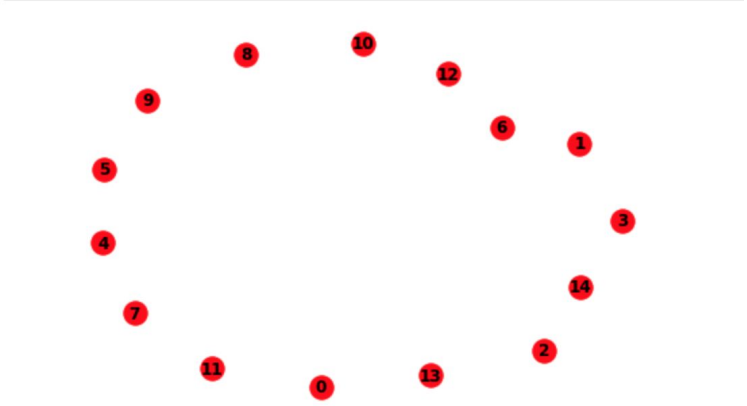

Fig 6: Unconnected Graph

Analysis

Intuitively, it seemed that there would be significant patterns of influence within all of the player's skills. For example, it makes sense that a player's weight should affect their aggression levels, that a player's height and weight should both influence agility and that a player's overall pace should affect their sprint speed. However, we did not observe any of these factors in our final Bayesian networks. Perhaps a contributing factor to this result is the fact that we ran our algorithm on a subset of ~1000 players, coming from a data set that featured 21,600 players, due

to limitations on computational resources. Additionally, the patterns of influence that we hoped to see were just guesses; maybe it is more accurate that most characteristics of a player are independent of each other. This would imply that every player is unique in their own way, and perhaps we cannot make a model to entirely generalize and capture patterns of influence.

Overall Conclusion

This project combined three different approaches - creating a predictive model based on player features, applying this predictive model to a betting agent modeled as a Markov decision process with two different incentives - maximizing profit and ensuring a threshold profit, and using Bayesian networks to determine relationships between features (in our case using the K2 algorithm to do so). Due to the nature of our problem (players being dynamic and thus their data not being relevant for long), we had the constraint of limited data to deal with, which is likely why methods that are typically successful such as neural networks lost out to less commonly successful predictive methods like random forests and logistic regression. As well, our Bayesian network analysis showed that there was little correlation between each of the numerical features we used to train our model, showing that each of the features we used to train our model was relatively unique. Finally, our simplified betting model did not take into account all the complexities of betting in the real world, but was able to come to similar conclusions to theory with respect to optimal policy actions based on win percentages. Expanding the model and state space to consider betting on all matches at once per round and increasing the number of possible actions would be great next steps to provide even further insight.

References

Caley, M. (2014, March 05). What is the best way to predict football matches? Retrieved from https://cartilagefreecaptain.sbnation.com/2014/3/5/5473358/what-is-the-best-method-for-predicting-football-matches

ArXiv, E. T. (2018, July 11). Machine learning predicts World Cup winner. Retrieved from https://www.technologyreview.com/s/611397/machine-learning-predicts-world-cup-winner/

How To Predict Football Results - bettingexpert. (n.d.). Retrieved from https://www.bettingexpert.com/how-to/predict-football-matches

Kelly, J. (1956). A New Interpretation of Information Rate. *IEEE Transactions on Information Theory,2*(3), 185-189. doi:10.1109/tit.1956.1056803

Sewell, M. (n.d.). Support Vector Machines vs Artificial Neural Networks. Retrieved from http://www.svms.org/anns.html

Featuring comprehensive current and historical squad details for clubs and national teams from all across the world. (n.d.). Retrieved from http://www.footballsquads.co.uk/

Kochenderfer, M. J., & Amato, C. (2015). *Decision making under uncertainty: Theory and application*. Cambridge, MA: The MIT Press.

Contributions

Gita Krishna (gitakris):

For this project, I spearheaded the background research component by analyzing previous works on this subject. Next, I worked on Part A of the project, which involved implementing a web scraping algorithm, feature engineering, formatting our data appropriately for the predictive models, and then implementing each of the four models described in Part A, as well as error analysis and conclusions on these models. Additionally, I implemented significant parts of the Bayesian Scoring function for Part C.

Kuhan Jeyapragasan (kuhanj):

Though I helped out with all three components, I primarily worked on part B of the project, modeling and implementing the Markov decision process to suit the gambling component of this project. I conducted prior research on optimal betting methods, specifically for knock-out tournaments and compiled betting data over the last 15 FA cups to compare bet-makers' odds with the odds given from our predictive model. I solved the Markov decision processes using a policy iteration algorithm since the transition probabilities between all possible states and the reward function were clearly mapped out in the construction of the problem, allowing for an offline policy like policy iteration to determine the best course of action.

Madhu Karra (mkarra):

I helped on various aspects of all three parts of this project, but focused mainly on part C. I worked on data collection, parsing and cleaning for the FIFA dataset and worked on feature engineering for K2. I worked on implementing various aspects/modifications to the K2 search algorithm specifically suited to our data. I also worked on collecting previous research and methods focused on betting methods and approaches towards the Bayesian network.