Association Football
Prediction of Team Squads and Positions

Deeksha Arya
McGill ID: 260786737
deeksha.arya@mail.mcgill.ca

Dipanjan Dutta
McGill ID:
dipanjan.dutta@mail.mcgill.ca

Anand Kamat
McGill ID:
anand.kamat@mail.mcgill.ca

December 21, 2017

Abstract

Association football receives great attention from fans worldwide. For popular leagues and championships, fans eagerly wait for the announcement of the playing 11 for a team. To this day, statistics of performance of players and teams are released, yet insight into how the selection process actually occurs remains confidential. In our project, we intend to develop an efficient machine learning algorithm to be able to predict the 11 on-field starting players for an upcoming match.

1 Introduction

Association football, or soccer, has a large player base in the world as well as a huge fan following. Estimated to have over 250 million fans as of the beginning of this century and a fan base of an estimated 1.3 billion [1], football attracts a lot of attention throughout the year. Championships and leagues hold interest and are also the base for large financial gains based on match outcomes and player performances based on bets. This is the motivation for attention to football in the machine learning world. Great enthusiasm exists worldwide in attempting to make predictions on all sorts of criteria. As match outcome is usually the most invigorating prediction, many scientists have done research on how to predict the result of a match given a team's previous performances.

Football is played between two teams of 11 players each. However, a team usually comprises of a total of 24 players, 5-7 of them are assigned for substitutions on match day and the remaining players are rested for the day and are not part of the match. A team of 11 will always have 1 goalkeeper, the other 10 will have assigned roles of attacker (try to score more goals), defender (try and prevent the opponent team from scoring goals) or midfielder (hold the major playing area and try to create chances for the attackers). The notation is given by 3 numeric digits indicating the number players of one of the roles mentions above on the field. For example, 4-3-3 indicates 4 defenders, 3 midfielders and 3 attackers being played. Of the squad of 24, very few team managers reveal how the selection of the Playing XI on any given day is made. In this project, we attempt to predict the Playing XI among a set of players given their statistics using a combination of machine learning classification and regression algorithms.

2 Problem Representation and Solution Methodology

In our report, we use the term *squad* to refer to the entire set of available players including reserves and substitutes.

Our approach to team prediction was divided into 3 main phases:

1. Classification of a player into position on field
2. Calculation of overall ranking of player
3. Determining best player in each position and placing them on the field

The idea for the first and second phase of the project was to create a learning algorithm which generalizes well to test data. In order to determine the best algorithm for these tasks, a number of algorithms were implemented, and using cross-validation, the best performing algorithm was then determined for the requirement.

2.1 Classification of a player into field position

Players must be classified into 1 of 4 positions on the field:

- Attacker
- Midfielder
- Defender
- Goalkeeper

Classifying a player into their roles and positions can be done efficiently using any ensemble classifiers. The two criteria for quantifying the role of a player (defender, midfielder or attacker) are the attacking and defending work rate. Assigning work rate of the player based on attributes and old data gives an effective metric that can further be used to quantify the roles using basic conditional statements.

2.2 Calculation of overall ranking of player

The calculation of overall ranking of a player is a pure regression problem. Given a set of features about a player, the algorithm must be able to successfully rate a player on a scale of 1 to 100. In order to do so, we began by trying to determine the best learning algorithm in terms of performance. We compared the results of five regression based algorithms namely:

- Linear Regression
- Ridge Regression
- Lasso Regression
- Decision Trees
- Random Forests

We began with the basic algorithm of Linear Regression, with the suspicion that maybe calculating the overall performance of a player could be a simple linear equation task on their attributes. In order to ensure no over-fitting, we also explored Ridge and Lasso Regression.

2.3 Selection of players in the starting XI

Given a team roster, the players can be classified into their roles. With a chosen formation, the top n players (players with best overall ranking) in every category are selected, n being the number of players of that role in the given formation.

3 The European Football Dataset

The dataset that we chose to work upon was the European football from Kaggle, created and curated by Hugo Mathien[2]. It contains data from seasons 2008 to 2016. Some of the data that is pertinent to our project is :

- Details of over 10000 players who are playing in the European leagues
- Match scores and results of over 25000 league matches.
- Players and Teams' attributes sourced from EA Sports' FIFA video game series, including the weekly updates

We used three tables from the dataset :

- Player data
- Player attributes
- Match data

3.1 Player data

This table 1 contains player IDs, and some basic information. The column metadata for the columns we used are as follows:

- **player_api_id** : Identification key for a player. Used to reference the player's attributes.
- **player_name** : Name of the player
- **player_fifa_api_id** : Identification key of the player in the FIFA video game database
- **birthday** : Birthday of the player
- **height** : Height of the player (in cms)
- **weight** : Weight of the player (in pounds)

Figure 1 shows five sample rows of this table.

player_api_id	player_name	player_fifa_api_id	birthday	height	weight
505942	Aaron Appindangoye	218353	1992-02-29 00:00:00	182.88	187
155782	Aaron Cresswell	189615	1989-12-15 00:00:00	170.18	146
162549	Aaron Doran	186170	1991-05-13 00:00:00	170.18	163
30572	Aaron Galindo	140161	1982-05-08 00:00:00	182.88	198
23780	Aaron Hughes	17725	1979-11-08 00:00:00	182.88	154

Figure 1: Player Data

3.2 Player Attribute Data

This table contains the player attributes and overall score of the player. For every player, there are multiple rows with different dates, indicating attribute updates of the player with progressing days. The updates range from the seasons 2008/09 to 2016/17. Player attribute updates can come from a variety of sources (FIFA database update, public football dataset update, match ratings etc.). The table contains a total of 42 columns, hence it is not possible to list all the columns here. We list some of the columns from the table below :

- **player_api_id** : Identification key for a player.
- **player_fifa_api_id** : Identification key of the player in the FIFA video game database
- **date** : Date of the update.
- **overall_rating** : Overall rating of the player in the FIFA video game database
- **potential** : Maximum overall rating the player is predicted to achieve.
- **attacking_work_rate** : Work rate of the player in the upper (attacking) half of the field. Factor variable with three levels : high, medium and low.
- **defensive_work_rate** : Work rate of the player in the lower (defending) half of the field. Factor variable with three levels : high, medium and low.

Of these, we extract the 34 numeric statistical attributes for every player, omitting attacking_work_rate and defensive_work_rate from the dataset. All attributes are integer values between 0 and 100. Some of them are listed below :

- **crossing** : Crossing ability of the player.
- **finishing** : Scoring and finishing ability of the player.

- **agility** : Metric to measure the running speed of the player.

There are several other metrics that fully quantify the player's abilities (including goalkeeper attributes). For our purpose, we extracted only those rows that contained valid values for all the dimensions of interest to us. As a result, a total of 183142 complete data rows were applicable to this project.

Figure 2 shows five sample rows of this table.

player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	crossing	finishing	agility
158023	30981	2012-02-22 00:00:00	94	96	left	high	medium	85	93	94
158023	30981	2011-08-30 00:00:00	94	96	left	high	medium	85	92	94
20801	30893	2015-10-16 00:00:00	93	93	right	high	low	82	95	90
20801	30893	2015-09-25 00:00:00	93	93	right	high	low	82	95	90
20801	30893	2015-09-21 00:00:00	93	93	right	high	low	82	95	90

Figure 2: Player Attributes

3.3 Match data

This table contains the data for matches played between European clubs from season 2008/09 to 2016/17. It contains starting players and substitutions in the games as well as betting odds from various websites. Some of the columns in this table are :

- **league_id** : Identification key for the league.
- **date** : Date of the match
- **home_team_api_id** : Identification key of the home team
- **away_team_api_id** : Identification key of the away team
- **home_team_goal** : Goals scored by the home team
- **away_team_goal** : Goals scored by the away team
- **home_player_1...11** : Starting player ID for the home team where home_player_1 refers to the goalkeeper
- **away_player_1...11** : Starting player ID for the away team where away_player_1 refers to the goalkeeper
- **home_player_X1...X11** : Substituted player ID for the home team
- **home_player_Y1...Y11** : Substituting player ID for the home team
- **away_player_X1...X11** : Substituted player ID for the away team
- **away_player_Y1...Y11** : Substituting player ID for the away team

There are several other columns but they are not important to our problem. This table 3 is used to separate the goalkeepers from the other players in the squad by filtering out the **home_player_1** and **away_player_1** player IDs and separating them from the **Player Attributes** table. It is to be noted that the team composition data is not present for all the matches. However, it is from experience we assume that a professional goalkeeper never changes his role in the team to another position and vice-versa.

league_id	date	home_team_api_id	away_team_api_id	home_team_goal	away_team_goal	home_player_1	away_player_1
4769	2015-11-28 00:00:00	9941	9831	2	0	698273	352860
4769	2015-12-05 00:00:00	9941	8689	2	3	698273	210114
4769	2015-12-19 00:00:00	9941	8639	1	1	698273	25540
4769	2016-01-16 00:00:00	9941	9847	0	1	698273	145550
4769	2016-01-30 00:00:00	9941	9747	1	2	698273	120598

Figure 3: Match Data

4 Algorithm Selection and Implementation

In this project, we focus on comparison of different machine learning algorithms with Decision Trees and Random Forests for both our classification and regression tasks. Our motivation for attempting to use Decision Trees as opposed to Deep Learning Neural Networks was mainly due to the clarity of the model. Neural Networks, though proven to be efficient algorithms when large datasets need to be learned, suffer from overfitting. Additionally, it can be computationally infeasible to train them on a larger dataset. A Decision Tree, on the other hand divides data into multiple parts with least loss of data, making it easier for us to understand and analyze the prediction mechanism. Injecting randomness in a decision tree mitigates the overfitting issue without compromising much on the performance.

4.1 Decision Trees

A Decision Tree is an upside down tree with its root node at the top and branches below it. At each node in the tree, a condition exists, based on which the tree branches further. The last nodes, or leaves, of the trees are ones from which no further branches exist.

During the training phase of a Decision Tree model, the tree is built, i.e. the conditions for splitting are determined. This along with the prediction mechanism differs based on whether the problem is a classification model or a regression model. We discuss both below [3].

4.2 Classification Trees

In the case of classification trees, the branching condition is determined by the Information Gain (IG) of a split.

When an attribute A splits a dataset D into smaller datasets D_i, we compute the average entropy of the split and compare the sum with the entropy (E) of the original set.

$$E(S) = -\sum_i p_i \log p_i \tag{1}$$

The Information Gain is then given by:

$$IG(S, A) = E(S) - \sum_i \frac{|S_i|}{|S|} E(S_i) \tag{2}$$

While building the tree, the model calculates the Information Gain from different possible split mechanisms, and ultimately chooses the condition of split that maximizes the Information Gain at that node.

Once the entire tree is generated, the model is ready to make a prediction. A test input is then logically traced through the tree until it reaches a leaf node. The majority class at this node is the prediction made by the model.

4.3 Regression Trees

Regression trees are different from classification trees because of their methodology for choosing the split condition at each node. In this case, instead Information Gain, the tree aims to maximize the Standard Deviation Reduction (SDR) at each node.

Standard Deviation Reduction (SDR) is given by

$$SDR(S, A) = SD(S) - \sum_i \frac{|S_i|}{|S|} SD(S_i) \tag{3}$$

where SD is standard deviation of a set

In the prediction process, the test data is once again, traced through the tree to a leaf node, where ultimately, the average of the values in this group is the prediction.

Decision Trees in general, have a great tendency to overfit. Early stopping mechanisms and pruning techniques are used to reduce variance of the tree. However, we find that even so, decision trees tend to be weak learners, and hence we explore Random Forests,a bootstrap aggregation technique to improve performance of Decision Trees.

4.4 Random Forests

Random Forests are combinations of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all the trees in the forest. After a large number of trees are generated, they vote (score) for the most popular class [5].

Decision trees, in general, suffer from overfitting and have low bias but high variance. The Random Forest algorithm mitigates the problem of overfitting to a great extent by "averaging" over multiple decision trees created from various subsets of the data. Although this induces bias in the model, it is highly accurate in predictions. The 'randomness' injected in the Random Forest model leads to minimizing the correlation between data points while maintaining strength. The algorithm also proves to handle noisy data quite well and is fairly robust to outliers [5].

This "averaging" of multiple decision trees is done by *bootstrap aggregation* (also known as bagging)m[6]. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by taking the majority vote in the case of classification trees . Because of the bootstrap aggregation, the algorithm handles overfitting by decreasing the variance while keeping the bias relatively low. Creating a model using only a single tree makes it prone to noise. Taking an average of multiple trees, over different datasets, reduces noise sensitivity (as long as the trees are not correlated). Taking a subset of attributes to train a decision tree eliminates the possibility of creating correlated decision trees (or same tree over and over again) [6]. The number of trees (B) is the only parameter of the model. We can decide on the optimal number of trees in the forest by using cross-validation.

5 Experimental Setup and Preliminary Test Results

5.1 Classification of a player into field position

Classification is the recognition of the category labels of instances that are normally described by a set of attributes (features) in a dataset. The aim of classification is to accurately predict class labels of instances whose values of attributes are known, but labels of classes are unknown.

The classification task of this project was to classify the players into one of the following four categories: Attacker, Midfielder, Defender and the Goalkeeper. The dataset used has two features, attacking work rate and defending work rate, which specified the player's affinity towards an attacking gameplay, which is common among attackers, or towards a defending gameplay, as commonly observed among defenders. We split the dataset into two parts : we trained the models on player data from 2008 to 2015, with three fold cross-validation, and kept the 2016/17 season player data as a test set. A supervised multi-class classification model can classify the attacking work rate and the defending work rate into the following category labels: high, medium and low. We experimented with several classifiers such as Naïve Bayes, multi-class single Neural Network (NN) and random forest.

Naïve Bayes is a very simple classifier and it fails to show significant improvement in performance as amount of training data increases. We trained the classifier on the training set using three-fold cross validation. Each data point consisted of 33 features, those discussed in the Player Attribute Dataset description 3.2 without the inclusion of *potential* of the player. The model was tested on a test data which gave an accuracy score of 63%.

Artificial Neural Networks (ANNs) are inspired by biological neural networks. ANNs learning show progressive improvement in performance and are suited for tasks involving a large set of data. Preprocessing involved centering and scaling the training data. A multi-class single layer Neural Network(NN) with a softmax activation function was used to classify the player data. We set the decay parameter to 0.1 and set a maximum weights allowed to be 20. These parameters were assusmed to limit the running time. The neural network performed better than the Naïve Bayes

classifier on the same training and testing data sets with an accuracy of 72%. Although this model shows a significance improvement over the simplistic Naïve Bayes classifier, we observed the model overfitted causing its test accuracy to plateau at 72%.

We also implemented the random forest classifier mentioned in section 4.4. The number of trees parameter (B) was set at 500. The random forest classifier, using three fold cross validation, performed significantly better than both the Naïve Bayes and the Neural Network(NN) classifiers with an accuracy of 85% on the test data.

The prediction accuracy of all the classifiers on the 2016/17 season data is shown in Figure 4:

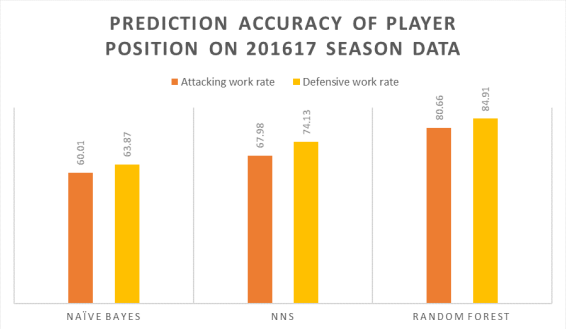


Figure 4: Prediction Accuracy of classifiers on 2016/17 season data

Being the best performing, the prediction results from the random forest classifiers were used to label the attacking work rate and the defensive work rates of the players. The following conditions were used to classify the players based on their predicted labels:

- Attacker: high attacking work rate and low defensive work rate
- Defender: low attacking work rate and either a high or a medium work rate
- Midfielder: all remaining categories

Classifying a player as a goalkeeper was quite straightforward as the dataset contained the features corresponding to goalkeepers for all players such as 'goal_keeper_reflexes'. All that needed to be done was separate players showing significant values in the 'goalkeeping' attributes.

5.2 Calculation of overall rating of player

The supervised regression algorithm used overall rating from the Player Attribute Dataset as the target output. Each input data row comprised of 34 features as explained in Section 3.2.

Our first approach to training a regression model for this task was to divide the available dataset into training and testing in the ratio 80:20 respectively.

The five algorithms namely Linear Regression, Ridge Regression, Lasso Regression, Decision Trees and Random Forest Regression were implemented using the sklearn[4] libraries with the default configurations.

To determine the best regularization parameter for Ridge and Lasso, 5-fold cross-validation was performed for the parameter values ranging from 0.001 to 1 with a multiplicative difference of 10. Figure 5 shows that the algorithms perform the best with a low regularization parameter of 0.001. This value was then used to implement the same algorithms and compare the error rate with the other regression algorithms.

As can be seen in the figure 6, Random Forest Regression tended to perform the best among the 5 discussed algorithms. We then ran the Random Forest Regression model on the test data and obtained an impressive low mean squared error of 0.94.

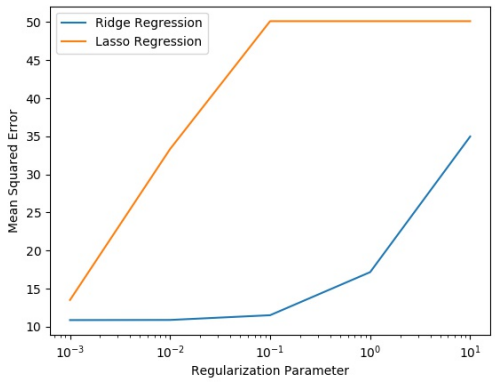
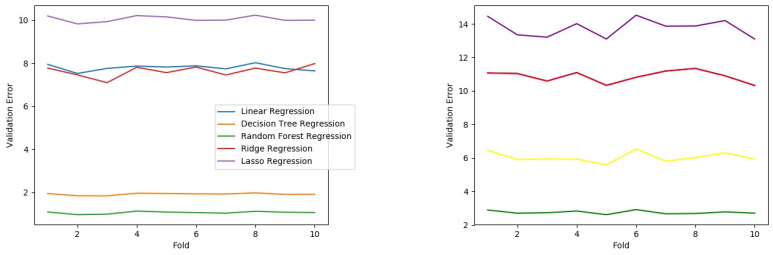


Figure 5: Error with respect to regularization parameter

However, upon analysis, we realized that the training-test split may not capture real world data well. Over time, it is likely that the means by which overall ranking is calculated changes over time. A random sampling of test data could not sufficiently determine if with current player attributes, the model is appropriate or overfits or underfits. In addition, we realized intuitively that the *potential* value of a player would likely be an evaluation made using the overall rating, and not be a feature that could contribute to predicting this player score. Hence, for our next experiment, we omitted this dimension.

Hence, we then separated player attribute data into details from seasons 2008-2015 and 2016/17. These were treated differently as train and test data respectively. The intuitive choice for this divide was also to determine whether over time, there was a drastic difference in scoring mechanism. If this was so, though training error would be low, test error would be high. If thought about analytically, this would not be because of overfitting, but more because the 2016/17 data could have been an entirely different rating mechanism, hence resulting in the entire cluster of data to lie beyond any previous model trend.



(a) Training data: 80% of total data (b) Training data: from 2008-2015

Figure 6: 10-fold cross validation

Here, it is important to note that in both data sets, there may be a repetition of player identities. For example, a player X may have two entries for different seasons, with different statistics and possibly a different overall ranking. We ignore the player identity and simply treat them as independent examples in the training and testing process.

The algorithm with the best cross-validation performance was then picked to predict on previously unknown data.

From the graph 6, it seems that Linear Regression curve has not been plotted, however this is not the case. The Lasso Regression and Linear Regression values are so closely matched that their curves coincide. Hence, the application of Lasso Regression seems fruitless in this case. It can be concluded that the Linear Regression algorithm does not perform well on this dataset. The issue is not overfitting as both Lasso and Ridge regression, perform better as *lambda* decreases 5, indicating the model's tendency to nullify the regularization term. Hence this model seems to underfit, and the dataset is likely non-linear in nature.

The Decision Tree Regression algorithm though performs better, still produces an error of about 6.03%. Though this is nearly 50% better than the results obtained by linear mechanisms, it seemed as though accuracy could potentially be improved. The Random Forest Regressor, performed the best of all algorithms with a cross-validation average error of 2.75.

We then ran the Random Forest Regression model on the test data and obtained an impressive low mean squared error of 0.79.

In both our above experiments, Random Forest Regression performed with least error during cross-validation. As a result, this was chosen as the machine learning model to predict the ranking of a new player.

5.3 Selection of players in starting XI

The input to the model will be the roster for a European club and the preferred formation. Once all the players are given, the classification model classifies the players into their roles (goalkeeper, defender, midfielder or attacker). The regression model calculates the overall rating for each player on the roster. With the information, the starting 11 players are selected from the roster according to the formation. Thus, if the formation is 4-4-2, the algorithm outputs the top goalkeeper, top 4 defenders, top 4 midfielders and top 2 attackers for the team from the given roster.

6 Predictions and Observations

After training and testing the above models on the 2016 season data, we decided to use the random forest classification and regression models to predict the starting XI of a football team given their current squad and preferred formation. We chose Arsenal as our team of choice. The current roster for the Arsenal 2017 season is :

- Aaron Ramsey
- Alex Iwobi
- Alexandre Lacazette
- Alexis Sanchez
- Calum Chambers
- Danny Welbeck
- Francis Coquelin
- Granit Xhaka
- Hector Bellerin
- Jack Wilshere
- Laurent Koscielny
- Mathieu Debuchy
- Mesut Oezil
- Mohamed Elneny
- Nacho Monreal
- Olivier Giroud
- Per Mertesacker
- Petr Cech
- Santi Cazorla
- Sead Kolasinac
- Serge Gnabry
- Shkodran Mustafi
- Theo Walcott

2 We retrieved the latest player attribute data from the dataset for each of these players and passed these details through the classification and regression tasks. The first model identified the position of each player, and the second then determined the overall score for each player. After which, the best possible lineup for Arsenal, if they played a 4-3-3 formation, was predicted. Figure 7 shows the lineup predicted by our algorithm along with the overall scores of the playing XI.

In order to evaluate the quality of our prediction, we compared every player in our prediction set to their performance in matches in which they have appeared this season. As a player is essential in a team's performance, we intuitively assess based on the number of matches won when the player is a part of the Playing XI. These results have been tabulated in Figure 8. This season, Arsenal has so far had a low season, winning only 63% of matches the team has played. Hence, it is clear that every player selected by the algorithm has a significant impact to the results of the game, performing approximately at the level of the team. This is not withstanding other factors like the

Figure 7: Predicted Arsenal 2017 lineup (made using BuildLineup)

performance of the other team members and non-measurable factors such as team chemistry. Our algorithm does not take into account the injuries of a player in the season, hence Santi Cazorla is selected because he is a highly rated player but he has not played a single game this season due to injuries.

7 Conclusion

The project explores the use of machine learning techniques to predict the best Playing XI for a squad, given the formation. This research is driven by the overwhelming increase in the pool of available sports data in European football and the dearth of team composition prediction models. Predicting the optimal playing team has lots of challenges, hence a forecasting model focusing on the team selection can be groundbreaking. In this project, we implemented an ensemble classifier to identify the best position for a player with a given set of attributes. We also, calculated the overall

Player	Appearances EPL	Appearances League Cup	Appearances Europa League	Wins	Goals	Clean Sheets	Win Percentage
Petr Čech	18	0	1	11	0	10	53%
Shkodran Mustafi	9	0	1	7	1	-	70%
Laurent Koscielny	15	0	0	9	0	-	60%
Per Mertesacker	4	1	3	5	1	-	62.5%
Francis Coquelin	6	1	4	7	0	-	64%
Santi Cazorla	0	0	0	0	0	-	N.A.
Mesut Özil	15	0	0	8	3	-	53%
Alexis Sánchez	15	1	1	11	5	-	73%
Jack Wilshere	7	2	6	10	1	-	66.67%
Alexandre Lacazette	18	0	0	11	8	-	61%
Olivier Giroud	15	2	6	13	7	-	56%

Figure 8: Team performance for selected players for 2017/18 season

rating of a player using regression trees. Finally, we created the Playing XI of a given formation using the players in the team's roster. The ultimate output of the model is highly relevant, and hence performs well in predicting the Playing XI for a football match from a squad.

8 Discussion and Future Work

Team composition prediction can be very useful in a lot of circumstances. Managers can prepare and strategize their own team based on their prediction of opposing teams. This prediction is also a very informative and useful tool for on-line fantasy football and betting games. A probable team might give insights to bidders on how to place bets or create their roster. There can be several possible extensions and improvements to the project.

- Including other data sources such as match commentaries and expert reviews should ideally improve the overall score estimate of a player and add some real-life bias instead of depending on pure statistics.
- Context information regarding head-to-head player statistics would be beneficial in predicting a playing XI against a given opponent team composition.
- Future work could be to predict specific field positions or areas for a player in its role (for example, a defender can be classified as Left Back, Center Back or Right Back) by gathering data on features such as team strategies and player performance in various positions.
- Insight into players frequency as substitutes and reserves could also help better assess their performance and rating.

  Most of the research focus, thus far, has been on predicting winners in a football match or tournament. Very little research and model development works towards the problem we present in our paper. Hence, this project is just the first stepping stone towards a robust team composition prediction model.

References

[1] Eric Weil, Peter Christopher Alegi, Jack Rollin, Bernard Joy, Richard C. Giulianotti, Football, `https://www.britannica.com/sports/football-soccer`, Last Updated October 2017

[2] Hugo Mathien, "European Soccer Database", `https://www.kaggle.com/hugomathien/soccer`, Kaggle, Last Updates October 2016

[3] J. Furnkranz, "Decision Tree Learning" `http://www.ke.tu-darmstadt.de/lehre/archiv/ws0809/mldm/dt.pdf`

[4] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research, Vol. 12, pg 2825–2830, 2011.

[5] Andy Liaw and Matthew Wiener, "Classification and Regression by randomForest", Vol. 2/3, December 2002.

[6] Leo Breiman, "Random Forests", Statistics Department, University of California, January 2001.