Soccer team formation and replacement system

Anusha Raman	Arunkumar Selvaraj	Karthiga Hariharan	Namitha Sudheendra	Pratik Sahoo	Srinath Ganesan
Master of Computer Science, ASU	Master of Computer Science, ASU	Master of Computer Science, ASU	Master of Computer Science, ASU	Master of Computer Science, ASU	Master of Computer Science, ASU
araman10@asu.edu	aselvar3@asu.edu	kharihar@asu.edu	nsudhee1@asu.edu	psahoo1@asu.edu	sganes25@asu.edu

Abstract **3**

Introduction **3**

Problem description **4**

Problem formulation 4

Algorithms/Models used 5

Random forest classifier 5

Logistic regression 6

Support Vector Machines 6

Naive Bayes 7

Random walk graph kernel 7

Methodology **8**

3.1 Data Cleaning and Pre Processing 8

3.2 Feature Selection using Random Forest 8

3.3 Team Formation 10

3.3.1 Predicting the Best Players using Logistic Regression, SVM and Gaussian Naive Bayes 10

3.3.2 Forming the best team 11

3.4 Player Replacement Strategy 13

Results **15**

Conclusions and future work: **15**

Conclusion 15

Contributions 15

Key learnings 16

Future work 16

Acknowledgements **17**

References **17**

Abstract

Forming an efficient and well performing team and excelling in every match and tournament is every team's ultimate goal. May it be any sports, the first and most important step is to form a well coordinated team. In today's world Data Science and Machine Learning are using historical data and coming up with solutions to real world problems.In this project we have proposed a team formation and team replacement model using Machine Learning Algorithms, where model takes in consideration various skill and performance parameters of each players, their role and puts it in various popular machine learning algorithms and comes up with the best players, which are then compared with a skill matrix and coordination matrix, which ultimately gives us the most efficient team. Also we have validated the team we got as an output with the original team composition of an actual football team's data over the course of a season. Further we have also implemented a Team replacement Algorithm, using graph kernel, to tackle unfortunate situations when a player gets injured or unavailable.

Keywords: Machine learning, Team Formation, Team Replacement, Soccer Team

1. Introduction

With the advent of data science and machine learning, no sport and no franchise has been immune to incorporating the benefits into the way a sports franchise is run from player recruitment, health, nutrition et al. Data driven approaches have made athletes and team better and fitter and are increasingly getting the acknowledgement of contributing to the fine margins that decide outcomes in top-level sports. Player and match statistics are collected every day and stored, consolidated and analysed. There are more and more attempts to reduce the intangibles in high level sports and to make everything quantifiable. Many novel metrics are being developed to not only analyse individual metrics but in game statistics and team dynamic metrics that are helping franchises in soccer work on tactics and strategies, areas that were purely the domain of the manger because nobody believed they could be quantified.

The emergence of fantasy X leagues (X = any sport, any competition) into a billion dollar industry has catapulted statistics into the mainstream where millions of users analyze match statistics to form their ideal team and try to be the best manager out there. Entire gaming industries (manager mode in FIFA by EA sports), football manager by Sports Interactive release iterations with improvements every year. These online fantasy league series and the computer/console games mentioned above are almost entirely driven by data.

Here, we attempt to see if a team we form using machine learning techniques effectively captures the underlying intangible ideas and opinions, that only the manager of the team knows about. We try to formulate a novel approach that takes into account the skill set of all the players in the squad and their coordination among them. We then evaluate this team with the one that

actually played real-time over the course of a season. We also try to find a replacement to a player if he becomes unavailable.

FIFA complete player dataset:
The dataset lists the elaborate characteristics of ~18000 players with 185 features. We have challenges to prune the feature space for derived features and ensure conditionally independence.

European matches dataset:
The dataset contains the summary of all matches played in Europe with all the participating players and results.

There have been attempts to formulate this `individual skill + team coordination (dynamics)` for team member replacement [cite paper]. We realized that the same concept could be extended to team formation too and attempted to do that. The former (team member replacement) hasn't been applied to sports.

2. Problem description

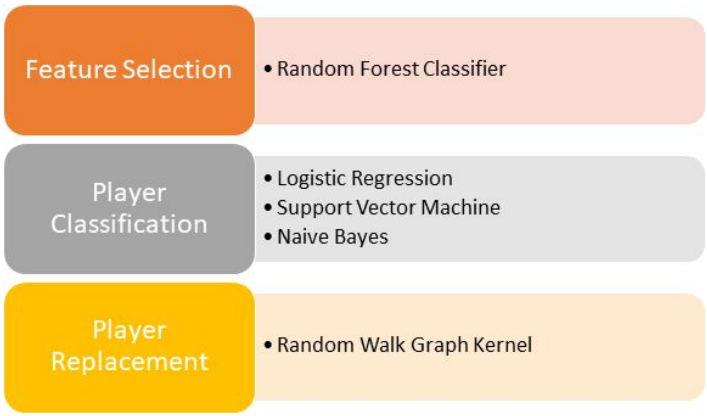
2.1. Problem formulation

Through this project, we try to form a soccer team for a club (Leicester city, 2015/16 season) using Machine learning models, that tries to effectively capture the skill set of the players available and the team dynamics. To achieve this, we use the FIFA complete player dataset (~18000 players and 85 features for each player) and the European matches dataset (all matches, teams, players who played, match results).

We then evaluate the team that we formed with the team that Leicester City fielded over 38 matches in the English premier league in 2015/16. Leicester city won the Premier league against all odds that season. The reason we chose this team in this particular season was to eliminate factors that we couldn't account for when evaluating. This was because, Leicester city had a very small squad and fielded the team in the same formation (4-4-2, 4-defenders, 4-midfielders, 2-forwards) throughout the season. Also, they fielded the most consistent starting 11 in every match compared to the other 19 teams.

This helped minimize quirks in managerial decision, player unavailability, fixture specific teams - data for all of which is hard to get and quantify. Under these assumptions, we tried to test the hypothesis we had and see if we could mimic their manager.

2.2. Algorithms/Models used



2.2.1. Random forest classifier

Random forests are a more popular machine learning method for feature ranking. They require a very little feature engineering, parameter tuning and it can work on both categorical and numerical variables as input.

A simple decision tree isn't robust. Random forest are an ensemble learning method for classification/ regression that operate by constructing multitude of decision trees at training time and outputting the class that is mode of the classes or mean prediction of individual trees. This model controls for overfitting and can often produce a very robust, high performing model.

The reason they are used for feature selection is because tree-based strategies used by random forests naturally ranks by how well they improve the purity of the node. This mean decrease in impurity over all trees (called gini impurity). Nodes with the greatest decrease in impurity happen at the start of the trees, while notes with the least decrease in impurity occur at the end of trees. Thus, by pruning trees below a particular node, we can create a subset of the most important features. From the Random Forest Classifier feature importance values, we can get a sense of list of variables that have the most effect in the prediction task. These information can be used to engineer new features, drop out features that look like noise and we can continue building our models. Sklearn libraries of these models have a ".feature_importances_" attributes which would return an array of each feature importance in determining the split.

Random Forests come with their own gotchas, especially when data interpretation is concerned. With correlated features, strong features can end up with low scores and the method can be biased towards variables with many categories. As long as the gotchas are kept in mind, and data is processed before running Random Forest classifier, this method would be the most useful method for feature selection.

2.2.2. Logistic regression

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Unlike actual regression, logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output is a probability that the given input point belongs to a certain class.

In Logistic Regression, the features are combined linearly using weights or coefficient values to predict an target value. Logistic Regression models the probability that an input (x) belongs to the default class (y = 1). Logistic Regression is a linear method, but the predictions are transformed using the logistic function. The coefficients of the logistic regression algorithm must be estimated from training data. Maximum Likelihood estimation is used for this purpose. It is a common machine learning algorithm that makes some assumptions based on the distribution of the data. The best coefficients would result in a model that would predict a value very close to 1 for the default class and a value very close to 0 for the other class. The intuition for Maximum Likelihood Estimation for Logistic Regression is that a search procedure seeks values for the coefficient that minimize the error in the probabilities predicted by the model to those in the data. A minimization algorithm is used to optimize the best values for the coefficients of our training data.

Logistic Regression assumes a linear relationship between the input variables with the target. In case of highly correlated inputs, the model can overfit and in such cases, the likelihood estimation process that learns the coefficients would fail to converge.

2.2.3. Support Vector Machines

SVM are supervised learning models with associated learning algorithms that analyze data used for classification & regression analysis. It is a non probabilistic binary linear classifier. It performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors that define the hyperplane are the support vectors. To define an optimal hyperplane, we need to maximize the width of the margin.

If the data is linearly separable, there is a unique global minimum value. An ideal SVM should produce a hyperplane that completely separates the vectors into two non-overlapping classes. However, perfect separation may not be possible, or it may result in a model with so many cases that the model does not classify correctly. In this situation SVM finds the hyperplane that maximizes the margin and minimizes the misclassifications. The algorithm does not minimize the number of misclassifications but the sum of distances from the margin hyperplanes.

SVM handles non linear data by using a kernel function (nonlinear) to map the data into a different space where a hyperplane (linear) cannot be used to do the separation. It means a

non-linear function is learned by a linear learning machine in a high-dimensional feature space while the capacity of the system is controlled by a parameter that does not depend on the dimensionality of the space. This is called *kernel trick* which means the kernel function transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

2.2.4. Naive Bayes

Naive Bayes is a collection of classification algorithms based on Bayes Theorem. It will assign the class labels taken from the finite set to the test data set. As per the Naïve Bayes algorithm, every feature being classified is independent of the value of any other feature. Each feature will contribute individually to the probability ignoring the correlation between the features.

In a classification problem, our hypothesis may be the class to assign for a new data instance (d). One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes theorem provides a way that we can calculate the probability of a hypothesis given prior knowledge. Class and conditional probabilities are calculated and predictions are made by bayes theorem using these values.

Advantages of Naïve Bayes is that it is relatively simple to understand and build the model, training requires only a small dataset, faster and insensitive to irrelevant features. But the features in a dataset aren't always independent, naïve bayes assuming it to be independent and predicting the probability is a shortcoming of this algorithm.

2.2.5. Random walk graph kernel

A Graph Kernel is a kernel function that computes an inner product on graphs. It can be intuitively understood as functions measuring the similarity of pairs of graphs. They allow kernelized learning algorithms such as support vector machines to work directly on graphs, without having to do feature extraction to transform them to fixed length real valued feature vectors.

There are many graph kernels - Sub-Tree Graph Kernels, Shortest-Path Graph Kernels, Optimal Graph Kernels. Among this, Random walk graph kernel provides superior performance. It conceptually performs random walks on two graphs simultaneously, then counts the number of paths that were produced by both the walks. This is equivalent to doing random walks on the direct product of the pair of graphs, and from this, a kernel can be derived that can be efficiently computed.

3. Methodology

3.1 Data Cleaning and Pre Processing

We started with FIFA complete dataset which contained information about all the players. These information included performance parameters, personal information about each players, their position preferences and so on. The dataset contained information of 18000(rows) players and 85(columns) features. We used the preference values to label the players as Goalkeepers, MidFielders, Forwards, and Defenders using SQL. We removed the derived features and preference features once we labelled each player with desired positions. We removed these features to bring down the number of features to 38 from 85. The added features include 4 binary features which are the players label, i.e. the role or position of the player.

3.2 Feature Selection using Random Forest

Once the dataset is pruned, we are now going to implement Random Forest to select or extract the most important features for each player type. This step is important as we will get an idea about the most important features to be considered for each player type i.e. a forward may have different attributes which contributes to its his performance than that of a defender or midfielder or goalkeeper. Hence feature selection step is very important to have an idea of the most important parameters for each type of players. Also feature selection reduces the computational time and complexity by just considering the important features and ignoring the less important features. We decided a threshold value of 80% to decide the important features for each player type. The selected features along with their contribution percent for each position is given below.

GoalKeeper -> 8 Features

[('gk_diving', 0.14474248477109466), ('gk_positioning', 0.1373909920366923), ('gk_reflexes', 0.1347541417334035),('gk_handling',0.1342712318991853),('gk_kicking',0.09179010389621757), ('heading_accuracy', 0.07338086737048452), ('dribbling', 0.059074595631526276), ('positioning', 0.04841202155687466)]

Forward -> 17 Features

[('finishing',0.13315449179311686),('sliding_tackle',0.09899333189890763),('standing_tackle',0.08638971942762011),('marking',0.07320827747889432),('interceptions',0.06198884623587282), ('heading_accuracy',0.05339064442103182),('long_passing', 0.0449331409441231),

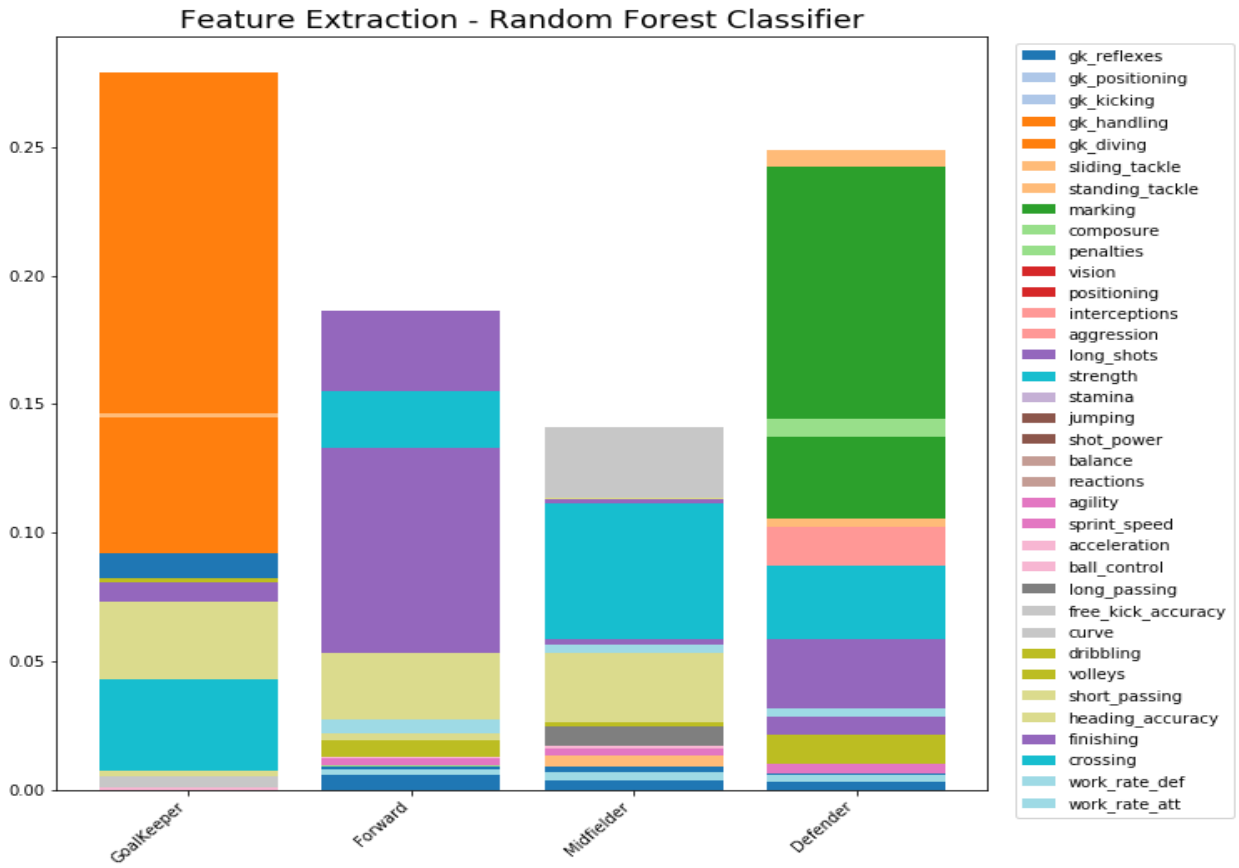
('penalties', 0.03804172354587509), ('volleys', 0.036402559200711934), ('positioning', 0.035056350204526264),('long_shots',0.03184154861452436),('crossing',0.022022341811360 78),('short_passing',0.019270285709448085),('dribbling',0.018107813639176594),('strength',0. 016819793116634164),('shot_power',0.016205066743603305),('free_kick_accuracy',0.0150328 43120874492)]

Midfielder -> 20 Features

[('long_passing', 0.09073878618893802), ('vision', 0.0895893997722711), ('short_passing', 0.05861063912302036), ('finishing', 0.058337450753178414), ('heading_accuracy', 0.05469550146367022), ('crossing', 0.052971325944738794), ('free_kick_accuracy', 0.05027573258659666), ('sliding_tackle', 0.04167597360794212), ('marking', 0.041060754005029836), ('long_shots', 0.03243507662816514), ('dribbling', 0.029226170034919164), ('standing_tackle', 0.02843906430012909), ('positioning', 0.028075893402961072), ('volleys', 0.026076956219611192), ('strength', 0.025310314362078812), ('curve', 0.024424862150089193), ('shot_power', 0.0207147238013438), ('balance', 0.019186866004694176), ('interceptions', 0.018608693964358974), ('penalties', 0.017727547973191694)]

Defender -> 15 Features

[('sliding_tackle', 0.14322440183717447), ('marking', 0.13712638961089657), ('standing_tackle', 0.10545608064493686), ('interceptions', 0.07560290423893982), ('vision', 0.058502281291068256), ('finishing', 0.05832411960573673), ('short_passing', 0.030510998700467844), ('long_shots', 0.03015008793367046), ('crossing', 0.028640748606360546), ('long_passing', 0.028250097140526634), ('aggression', 0.026550130076831675), ('volleys', 0.02515741593023758), ('heading_accuracy', 0.021175678650563363), ('positioning', 0.019755653488326686), ('penalties', 0.01944139066801634)]



3.3 Team Formation

3.3.1 Predicting the Best Players using Logistic Regression, SVM and Gaussian Naive Bayes

Now that we have the most important and differentiating features for each playing position, we train multiple machine learning algorithms including Logistic Regression, Support Vector Machine and Gaussian Naive Bayes to classify the players for the position they are best based on the skill value that he posses. The pool of 18000 classified players from the dataset was split into 80% training and 20% test data. The 80% split training data was used to train the aforementioned classification models. Once the models are trained, we used 20% split test data to test the accuracy of the build model. Also, the models were implemented not just to classify a player in one of the four playing positions but also to, give a probability value for each of the positions that tells how much skill he has for playing that particular position.

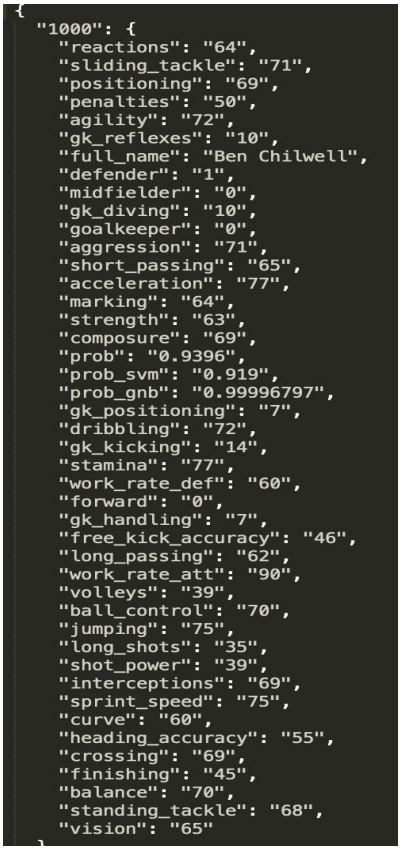
The pool of available Leicester City players are picked to test our trained model. The picked players are unclassified players about whom we have only the skills as features. When these

players are passed on to our model, we obtain 4 probability values for each of them about how good they are a match for Goalkeeper, Forward, Midfielder and Defender positions respectively. These probability values talk only about the skill of the player for the position. The players are then ranked based on the probability skill value obtained. Players with higher probabilities values are then considered for forming the team. A player's skill probability value along with his coordination value with other selected players using the coordination matrix is used in the next step to form an efficient and high performance team. The accuracies with which the probabilities for each player type is predicted using the 3 Machine Learning Algorithms are tabulated below.

Algorithm	GoalKeeper	Forward	Midfielder	Defender
LR	100	96	70	93
SVM	100	96	93	89
Naive Bayes	100	85	63	81

3.3.2 Forming the best team

From the skill matrix got from the previous step, we have the rank of players based on how good they are in their positions for the three algorithms - Logistic Regression, SVM and Naive Bayes.
So, we have the list of Goalkeepers, Defenders, Midfielders and Forwards, ranked in descending order of good they are for all 3 algorithms for Leicester city's squad of 27 players for the 2015/16 season. We also have a coordination matrix (27*27) that tells us how many times each pair of players played with each other.



```
{
  "1000": {
    "reactions": "64",
    "sliding_tackle": "71",
    "positioning": "69",
    "penalties": "50",
    "agility": "72",
    "gk_reflexes": "10",
    "full_name": "Ben Chilwell",
    "defender": "1",
    "midfielder": "0",
    "gk_diving": "10",
    "goalkeeper": "0",
    "aggression": "71",
    "short_passing": "65",
    "acceleration": "77",
    "marking": "64",
    "strength": "63",
    "composure": "69",
    "prob": "0.9396",
    "prob_svm": "0.919",
    "prob_gnb": "0.99996797",
    "gk_positioning": "7",
    "dribbling": "72",
    "gk_kicking": "14",
    "stamina": "77",
    "work_rate_def": "60",
    "forward": "0",
    "gk_handling": "7",
    "free_kick_accuracy": "46",
    "long_passing": "62",
    "work_rate_att": "90",
    "volleys": "39",
    "ball_control": "70",
    "jumping": "75",
    "long_shots": "35",
    "shot_power": "39",
    "interceptions": "69",
    "sprint_speed": "75",
    "curve": "60",
    "heading_accuracy": "55",
    "crossing": "69",
    "finishing": "45",
    "balance": "70",
    "standing_tackle": "68",
    "vision": "65"
```


To now form the team, we start with the premise that the best player for each position based on skill always gets selected into the team. So, we already have the goalkeeper, a defender, midfielder and a forward based on just the skill matrix. We have to pick the remaining 7 players from the 23 remaining players.

To do this, we pick each player, calculate a score for him based on the number of matches that he has played with the players selected so far and the weight associated with his skill value. We selected the player that has the highest score far and then, once selected, we remove him from the list of available players for the next iteration. This is repeated till all players are chosen, constrained by the formation (4-4-2 : 4-defenders, 4-midfielders, 2 defenders and 1 Goalkeeper).

To evaluate the chosen team, we find the percentage of players that we picked correctly for each match, for 38 matches and find the accuracy. This is repeated by varying the weights that we give to skill and coordination when choosing players 5-11. This is then repeated to evaluate the skill matrix got from all 3 algorithms. We found that we got the best results when the skill and coordination were weighted in the proportion 80:20. The accuracy for each algorithm for this proportion of weights is calculated and the accuracy is presented.

3.4 Player Replacement Strategy

Description

The goal of the replacement strategy is to reduce the impact of a player's departure, in case he becomes unavailable, by replacing him with another player who performs equally well in the team. The replacing player must have the same skills as the departing player - for example, a goalkeeper can never replace a defender. And, the arrival of the new player should not affect the coordination of the team.

Hence, the chosen player for replacement will have matching skillset and the best coordination with the team. To do so, we use graph kernel technique that measures similarity in the context of both skill and coordination.

Technique - Random Walk Graph Kernel

The approach used for Player Replacement is Random Walk Graph Kernel [**cite paper**]. For the above problem, we take as input the team of 11 formed earlier in this project. Let this team be represented as T and G(T) is a labelled graph for the original team under consideration.

Let P ϵ T be the member to be replaced and ¬ Q ϵ Tbe the replacing member. Then a labelled graph G(T(P → Q)) will represent the new team where P is replaced by Q.

Every graph G is a labelled social network represented as G := {A, L}
where, A is an n x n coordination matrix that characterizes the connectivity among the n different individuals; L is n x l skill indicator matrix in which the i^{th} row vector will correspond to the skillset of the i^{th} individual and l represents the total number of skills.

From the above definitions, we can find the replacement person Q as

$$q = argmax_{j,\neg j \in T} Ker(G(T), G(T_{p \to j}))$$

Where Ker (.) is the kernel between the two labelled graphs.

Random walk graph kernel is calculated as the Kronecker product between the two graphs as it gives the same result as the individual graph walks.

$$\ker\left(G(T), G(T_{p\to q})\right) = y'(Z^{-1} + cZ^{-1}X(I - cYZ^{-1}X)^{-1}YZ^{-1})\left(\left(\sum_{j=1}^{l} {L_1}^{(j)} \otimes L_C^{(j)}\right)x + \left(\sum_{j=1}^{l} ({L_1}^{(j)} \otimes e^{(j)})(I \otimes f^{(j)})\right)x\right)$$

where,
c = decay factor
X = Kronecker product

Z is precomputed as

$$Z = I - c\left(\sum_{j=1}^{l} {L_1}^{(j)} \otimes L_C^{(j)}\right)(A_1 \otimes A_c)$$

y = y1 x y2 x = x1 x x2 are the starting and stopping vectors indicating weights of different nodes

Algorithm

The below algorithm based on the Random Walk Graph Kernel approach [cite] finds the similarity score of for every available player in the squad who is not already present in the team T. The similarity score will denote how good a replacement he is for player P. The team manager then picks the player with the highest similarity score.

Algorithm 1: TEAMREP-FAST-EXACT

Input: (1) The entire social network $\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$, (2) original team members $\mathcal{T}$, (3) person p who will leave the team, (4) starting and ending probability $\mathbf{x}$ and $\mathbf{y}$(be uniform by default), and (5) an integer k (the budget)

Output: Top k candidates to replace person p

Initialize $\mathbf{A}_c, \mathbf{L}_1^{(j)}, \mathbf{L}_2^{(j)}, j = 1, \ldots, l$;

Pre-compute

$\mathbf{Z}^{-1} \leftarrow (\mathbf{I} - c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{A}_c))^{-1}$;

Set $\mathbf{R} \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})\mathbf{x}$; $\mathbf{b} \leftarrow \mathbf{y}^T\mathbf{Z}^{-1}\mathbf{R}$; $\mathbf{l} \leftarrow c\mathbf{y}^T\mathbf{Z}^{-1}$;

for *each candidate q in $\mathbf{G}$ after pruning* **do**

  Initialize $\mathbf{s} \leftarrow$ a zero vector of length t except the last element is 1;

  Initialize $\mathbf{w} \leftarrow$ weight vector from q to the new team members;

  Set $\mathbf{E} \leftarrow [\mathbf{w}, \mathbf{s}]$; $\mathbf{F} \leftarrow [\mathbf{s}'; \mathbf{w}']$;

  Set $\mathbf{e}^{(j)} \leftarrow$ a t by 1 zero vector except the last element is 1, for $j = 1, \ldots, d_n$;

  Set $\mathbf{f}^{(j)} \leftarrow$ a $1 \times t$ zero vector except the last element which is label j assignment for q;

  Set $\mathbf{P} \leftarrow [\mathbf{L}_1^{(1)} \otimes \mathbf{e}^{(1)}, \ldots, \mathbf{L}_1^{(t)} \otimes \mathbf{e}^{(t)}]$;

  Set $\mathbf{Q} \leftarrow [\mathbf{I} \otimes \mathbf{f}^{(1)}; \ldots; \mathbf{I} \otimes \mathbf{f}^{(t)}]$;

  Compute $\mathbf{X}_1 \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{A}_1 \otimes \mathbf{L}_c^{(j)}\mathbf{E})$;

  Compute $\mathbf{X}_2 \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{A}_1 \otimes \mathbf{e}^{(j)}\mathbf{f}^{(j)}\mathbf{E})$;

  Compute $\mathbf{Y}_1 \leftarrow \mathbf{Q}(\mathbf{A}_1 \otimes \mathbf{A}_c)$;

  Compute $\mathbf{Y}_2 \leftarrow (\mathbf{I} \otimes \mathbf{F})$;

  Set $\mathbf{X} \leftarrow [\mathbf{P}, \mathbf{X}_1, \mathbf{X}_2]$, $\mathbf{Y} \leftarrow [\mathbf{Y}_1; \mathbf{Y}_2; \mathbf{Y}_2]$;

  Update $\mathbf{M} \leftarrow (\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})^{-1}$;

  Compute $\mathbf{r}' \leftarrow \mathbf{Z}^{-1}\mathbf{P}\mathbf{Q}\mathbf{x}$;

  Compute score$(q) = \mathbf{b} + \mathbf{y}^T\mathbf{r}' + \mathbf{l}\mathbf{X}\mathbf{M}\mathbf{Y}(\mathbf{Z}^{-1}\mathbf{R} + \mathbf{r}')$;

end

Return the top k candidates with the highest scores.

[Reference Paper: Replacing the Irreplaceable: Fast Algorithms for Team Member Recommendation]

4. Results
5. Conclusions and future work:

Conclusion

The models used in this project include classification models, feature selection model and graph kernel technique. The conclusion with respect to every technique is given below:

Random Forest classifier model was used for feature selection. In general, when the dataset has correlated features, random forest will consider only one of them as predictor and hence the other features will lose importance. Data preprocessing was done in our dataset to remove the derived feature and hence Random forest gave us the actual importance of every feature in the dataset.

For team formation predicting best player task, we have used Naïve Bayes, SVM and Logistic regression Models. Naïve Bayes technique gave us the best accuracy, since our dataset is optimized using feature selection and it has a very less degree of class overlapping (targets are independent and there is almost no overlap). Compared to Logistic regression, ensemble model support vector machines performed well and gave better accuracy.

After getting the list of players for Leicester City's squad, ranked by how good they are for each position, we formed the team of 11 as mentioned in implementation. We then evaluated this playing 11 with the team of 11 that Leicester city fielded over the course of the season for the three algorithms and found that, everything else being the same, the skill matrix provided by Naive Bayes contributed to the best accuracy achieved in team formation, followed by SVM and Logistic regression.

Player replacement strategy correctly identified the best available replacement option for a given player with random walk based graph kernel algorithm.

Contributions

Team Member	Contribution	Percentage
Anusha Raman	Data Preprocessing, Feature Selection,Team Formation, Team Replacement	16.66%
Arunkumar Selvaraj	Data Preprocessing, Feature Selection,Team Formation, Team Replacement	16.66%
Karthiga Hariharan	Data Preprocessing, Feature Selection,Team Formation, Team Replacement	16.66%
Namitha Sudheendra	Data Preprocessing, Feature Selection,Team Formation, Team Replacement	16.66%
Pratik Sahoo	Data Preprocessing, Feature Selection,Team Formation, Team Replacement	16.66%
Srinath Ganesan	Data Preprocessing, Feature Selection,Team Formation, Team Replacement	16.66%

Key learnings

Some of the key learnings from this project that we as a team gained from this project were,

1. One of the main learnings from this project is that to perform a good team formation, we need to rank the players how good they are fit to be in a particular position. To choose them fit to be in a particular target, our classification models has to be trained with the features that is distinguishing every player to position. The importance of feature selection is realized as it contributes much to the accuracy of the classification models.
2. Existing work used skill and coordination values for the players to find a player that replaces another player. We realized the same can be used for the purpose of the team formation to come up with the best skilled and well-coordinated team.
3. The Team formation isn't a just a prediction/classification task. The learning we gained here was to use the probabilistic models that would not just classify a player to particular position but also give us the probabilities so that we can rank them and later use the highly skilled player to position with their coordination value to existing player for forming the best team.

Future work

- This system can be extended to take into account the wins/losses of a chosen team in order to strategically change the team for every match in the season.
- Complex team coordination metrics could be devised between each pair of players based on in-game statistics.
- This system can also be extended to take the recent form of the players into account for obtaining best results. That can be done by updating player's last match performance and calculating the skill matrix again before picking the playing 11 for the next match.

Acknowledgements

We would like to place our thanks to Dr.Hanghang Tong for patiently listening to our doubts and concerns and clarifying them in class and in-person when we required guidance for the approach we had decided upon for our project.

We would like to thank the Kaggle community for community sourcing of the datasets that we have used in our project.

References