The Ultimate Team

By Austin Lee and Drake Song

Introduction

The ultimate goal of this project is to objectively build the best team of soccer players using classification and regression techniques and to test team's ability using an accurate match predictor. If you ask a soccer fan to create their best team of 11 players, they will probably give you a subjective answer. Their choices can depend on the team they support, where they live, or their favorite players, so you will get different answers depending on who you ask. Our project aims to eliminate as much subjectivity in the process of choosing the 11 best soccer players.

To address the issue, we used three classification models and two regression models. We first created a match predictor model to test the ability of our ultimate team. Since we wanted to correctly determine the ability of our ultimate team, it was important to choose an accurate classification technique. Logistic regression was used since it outputs a probability rather than making a clear-cut decision, so it allowed us to set the threshold that would give a home win or home loss. The second step was to select the 11 best players for the ultimate team. SVM was used to decide whether a player would be on the ultimate team.

The problem we faced was determining the team attributes of the ultimate team and the also calculating the betting odds of the match. Our match predictor trained on the betting odds and team attributes of existing matches, so it required those inputs in order to make a decision. Since matches between our ultimate team and other teams never happened, there are no betting odds and since our ultimate team isn't actually a real team, there are no team attributes. To address this problem, we used Trees to calculate the betting odds and Bagged Trees to calculate the team attributes.With these attributes, we then utilized the match predictor model in testing out how well the ultimate team does against other teams.

Background

Logistic Regression

To build the match predictor, we used logistic regression. Logistic regression is a classification technique models the probability of two outcomes. In our case case, it outputs the probability of the home team winning and the probability of the away team winning. Usually, the strength of this method is that users can decide the threshold at which the regression classifies something as. For example, when using logistic regression to determine whether or not a patient has a condition, it's better to be safe than sorry, so a user might set the threshold lower than 50%. However, since the outcome of a soccer match is inherently random, we just set the threshold at 50%. One weakness of logistic regression is that it becomes inaccurate when you try to predict more than 2 classes. At first, we tried to predict 3 classes: win, lose, or tie, and this gave us around a 50% error rate. Then, we switched to just 2 classes: win or lose, which gave us an error rate of around 25%. In practice, the multiple-class extensions of logistic regression aren't used very often since there are more accurate methods.

Support Vector Machines

To predict the team of the year, we used support vector machines (SVM). The support vector machine is a natural choice for binary classification since SVM produces a non-linear decision boundary between the two classes. The performance of linear regression can suffer when there is no linear relationship between the predictors and outcome. SVM is an extension of the support vector classifier, which expands the feature space by using higher-order ($p > 1$) polynomial functions of the predictors. SVM expands the feature space in a specific way using kernels, which are functions that quantify the similarity of two observations. One can choose different kernels to use, for example, the polynomial kernel or the radial kernel. A weakness of SVM is that, for extremely large datasets, it requires extensive computational power and memory.

Regression Trees

To predict the betting odds to input into our match predictor, we used regression trees. Regression trees are built by dividing the predictor space into a certain amount of distinct and non-overlapping regions. For every observation that falls into a region, we make the prediction using the mean of the response values in the corresponding region. For example, if an observation x_1 belongs to a region R_1 and the response mean of R_1 is 10, then we will predict x_1 to be 10. The regions are constructed by finding the boxes that minimize the residual sum of squares, given by:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

However, using this method usually will overfit the data. *Pruning* is the process of growing a large tree and pruning it to get obtain a subtree, which will lead to it performing better on data outside of the dataset. Pruning can be computationally expensive since there is a large amount of possible subrees. A way of selecting a small set of subtrees to consider is known as cost complexity pruning. Cost complexity pruning depends on a tuning parameter that controls the subtree's complexity and its fit to the training data.

A strength of regression trees is that, since it can be displayed graphically, it's highly interpretable. However, for extremely large trees, it may lose some interpretability. The biggest weakness of regression trees is that they don't have the same level of predictive accuracy as other regression techniques and a small change in data can cause a significant change in the final tree.

Bagged Trees

Bagged trees was used to predict the team attributes for the ultimate team. Decision trees inherently suffer from high variance, so fitting two trees to different partitions of the same dataset will give different results. Bagging is a way to lower the trees' variance. The process involves generating different bootstrapped training data sets, training our trees on the bootstrapped sets in order to get a prediction, then averaging all of the predictions to get a final result. A disadvantage to using bagged trees is that it sacrifices interpretability, which was one of the best strengths of a decision tree. However, as a result, it improves the accuracy.

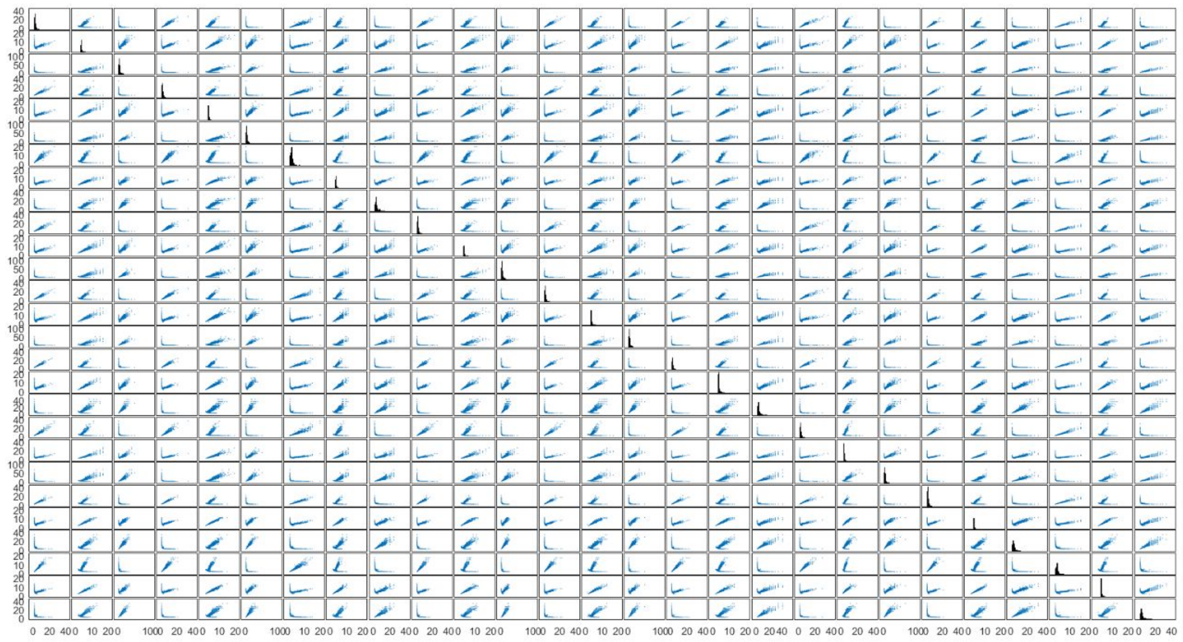
Methods

Predicting Match

For the team attribute dataset, there were three features: buildUpPlayPositioning, chanceCreationPositioning, and defenceDefenderLineClass, that needed to be converted into numbers.

For the match dataset, we created a result feature since the match only listed the amount of goals the home team had and away team had.

With any classification method, feature selection is an important step to improve the accuracy of the model. First, we looked for collinear features using collinearity plots.



This matrixplot shows correlations between all of the betting odds. Looking at the first column of this plot, it is clear to see linear correlation with every features except features 3, 6, 9, 12, …. Because these are betting odds from numerous top sports betting sites, these linear correlations are expected. While home team winning odds and match being tied odds are linearly correlated, home team winning odds and away team winning odds actually don't show linear correlation. For this reason, home team winning odds and away team winning odds from B365 (this site has the most number of betting odds available) are used for analysis.

Using all 24 features available (2 betting odds, 11 home team attributes, 11 away team attributes), the following p-values were outputted by the logistic regression model:

0	0.0340
1	5.3308e-54
2	6.6708e-62
3	0.6549
4	0.9522
5	0.3314
6	0.2074
7	0.6714
8	0.5010
9	0.1107
10	0.4469
11	0.3143
12	0.5431
13	0.3267
14	0.5683
15	0.1929
16	0.4385
17	0.0424
18	0.1321
19	0.0787
20	0.6590
21	0.1210
22	0.3730
23	0.4385
24	0.8385

After removing one feature at a time starting with the feature with the biggest p-value, five features were left: B365H (betting odds of home team winning), B365A (betting odds of away team winning), homeBuildUpPlayPositioning, awayDefencePressure, awayDefenceTeamWidth. The p-values produced by these features are:

0	0.0033
1	1.2378e-51
2	2.4913e-70
3	0.0185
4	0.0114
5	0.0302

These features were then used to create a logistic model that will predict the outcome of a match.

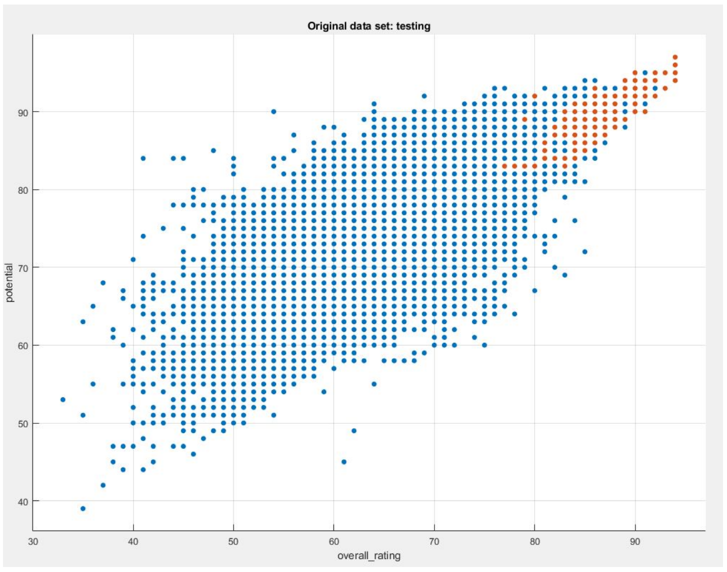
Creating the Ultimate Team

In creating the Ultimate Team, attributes of the players had to be cleaned first. Preferred_foot of each player was converted to either 0 for 'left' or 1 for 'right'. The dates on which the players' attributes were calculated and recorded has to be changed to a categorical 'season' feature to comply with the 'season' feature we had been using for match data.

From the official Union of European Football Associations (UEFA) website, Team of the Year (shorthand: TOTY; each year, millions of fans worldwide vote for the best 11 players and 1 coach for the year based on performance) data was extracted by hand. The extracted data contains 110 players who were selected for Team of the Year from year 2006 to 2016. Because there are multiple players who were selected multiple times, it was important to keep track of seasons as the player attributes differ according to seasons. Also, due to special characters, nicknames, and other string problems, some names had to be adjusted. From this list of players' names, we had to cross-check with the data we already had to obtain each player's api_id so that we would be able to accurately obtain the player's attributes without having to worry about more strings and special characters.

Going through the entire player attributes data which contains a player's attributes (36 features), we assigned 0 if the player in the particular season had not been selected for TOTY and 1 if the player had been selected.

Using player attributes as X and the binary categorical result as Y, we utilized a quadratic SVM model in creating a model that will help us determine if a player will be selected for TOTY or not. We tried linear SVM, quadratic SVM, cubic SVM, and Gaussian SVM, and quadratic SVM gave the best results. This is due to the nature of the data where only the top players are selected for TOTY.



The figure here is a simplified demonstration of the fact that quadratic SVM is best fit for this data. Graphing the overall_rating (higher the number, the better the player is) of a player against the potential (potential rating of a player; higher the better), it is clear to see that the player who have been selected for TOTY (the red dots) are all on the top right side of the plot where all of the highest rated players are at. Since the boundary between the red and blue points are not quite linear nor cubic, quadratic SVM gave us the best results.

For our test data, we found another data set that was recently made. This data set contains player attributes from the most recent FIFA18 soccer game that was released 3 months ago in September 2017. The player attributes we have been using were from previous FIFA games from FIFA06, FIFA07, …, FIFA17 so using FIFA18 as our test data was perfect. Although these are video games, these are officially licensed by Fédération Internationale de Football Association (FIFA) and closely follow and analyze real-time player and match statistics.

The FIFA18 dataset had to be cleaned a bit in removing unnecessary features that were not available in the previous dataset. The variable names also had to be changed to match the ones used previously. Player ID numbers also had to be converted because FIFA18 dataset used player_fifa_api_id (player's ID number within the game) instead of player_api_id (official player ID number) which we have been using.

Using FIFA18 dataset, we plugged it into the SVM model we created earlier to determine which player from FIFA18 will make it to the TOTY next year. After 100 runs through the model, we used the 11 most frequently appeared players for our Ultimate Team.

Cristiano Ronaldo	Forward
Lionel Messi	Forward
Manuel Neuer	Goalkeeper
David de Gea	Goalkeeper
Sergio Ramos	Defender
Gianluigi Buffon	Goalkeeper
Antoine Griezmann	Foward
David Alaba	Defender
Neymar	Foward
Toni Kroos	Midfield
Leonardo Bonucci	Defender

Predicting Features

Since the model that predicts the outcome of a match uses 5 features, we decided to use regression models in determining those features for the Ultimate Team. Putting the Ultimate Team as the home team for all the matches, we would need to create 3 models to predict 3 features: B365H, B365A, and homeBuildUpPlayPositioning.

In order to do so we had to modify our dataset. The original match dataset that we have contains the player ID for all 22 players (11 home and 11 away). For each player in a match, we extracted the player attributes for that player (35 features). This created 770 features for about 20k matches. Using this huge data, we put them into two Tree regression models with minimum leaf size of 4 (4 seemed to work out the best) in predicting the two betting odds. In determining the homeBuildUpPlayPostioning feature (0/1), we used Bagged Tree classification model as Bagged Trees gave us the best result.

From the FIFA18 dataset, we obtained the player attributes for the Ultimate Team. Using these player attributes, we replaced the home attributes from all of the matches and created a new match data set. With this new dataset, the three Tree models created to predict the three features needed for the predictMatch model were then used.

Testing the Ultimate Team

With the 3 features predicted from 2 regression models and 1 classification model and the 2 other features (awayDefencePressure and awayDefenceTeamWidth) extracted from existing dataset, we plugged this new match dataset into the predictMatch model we created earlier to see how well the Ultimate Team (home team) does against all of the other teams in about 20k matches.

Results

Predicting Match

The logistic regression for predicting the outcome of a match produced error of about 26.9325%. The confusion matrix produced is:

	1	2
1	688	93
2	238	210

Creating the Ultimate Team

The SVM model produced error of about 0.121366%.

Predicting Features

Model for B365H created RMSE of about 1.43536, B365A created RMSE of about 2.54195, and homeBuildUpPlayPostioning created error about 1.39809%.

Testing the Ultimate Team

The Ultimate Team won about 9550 matches which is 70.2929% of matches played.

Discussion

Predicting Match

At first, we were trying to predict if a match would result in home team winning, away team winning, or a draw. This turned out to be a problem because no matter which model we used, the error was greater than 50% and sometimes 80%. So we decided to just do home team win or away team loss since the most important games in soccer cannot end in a draw.

From the error rate produced it is clear to see that having to just predict 2 outcomes instead of 3 performs much better. Although it could be better high-20% error is acceptable. Looking at the confusion matrix, it is clear to see that the model does a decent job in predicting if the home team is going to win. On the other hand, the model does not do so well in predicting if away team is going to win. In fact, the model falsely predicts more of matches where the away team wins than it correctly predicts.

We were satisfied with the final accuracy of the match predictor and believed it was accurate enough so we could use it to determine the ability of our ultimate team. The betting odds are really crucial to the match predictor and probably the reason why the predictor is accurate in the first place. Each betting website probably uses some sort of data modelling to get their odds, so our logistic regression model is building a model on top of their model. Also, since our model takes into account some team attributes, it should be fairly accurate when deciding the outcome of a game.

Creating the Ultimate Team

As discussed earlier, this model should be a simple one due to the fact that all the players who were selected for TOTY are gathered up on the top right of the plot in simple terms. In fact, the quadratic SVM only produces 0.121366% error which is unbelievably good.

Predicting Features

The RMSE produced by the two regression models are definitely higher than we would’ve liked. Because both betting odds range from 1 to around 20, having RMSEs of 1.5 and 2.5 cause the resulting betting odds to be not too accurate. However, because this model produced the best results so far, we just had to work with them.

The Bagged Trees classification model gave us a very small error of only 2%. This is mainly due to the fact that there were only two classes: 1 and 2.

Testing the Ultimate Team

Because the predictMatch model does well in predicting if home team is going to be winning, the Ultimate Team was put in all of the matches as the home team. The result wasn’t the best however it was satisfactory with the amount of models and the data we had to work with.

Also we realized at the end that there was an underlying assumption we were looking over. We were assuming that high betting odds and high team attributes were correlated. In a perfect world, this would be the case; however, soccer matches in real world depends on not just player and team attributes but also on other factors such as location, time, weather, condition of the players (did the team just play in a tournament? Are the players exhausted from traveling? etc.), and so much more that mere team attributes and player attributes are most likely not enough to predict the outcome of a match.

Future Work

One possible step to address our original goal of objectively creating the best team would be to use introduce new features such as the amount of goals scored or minutes played by a player. Since our data didn't contain any field statistics such as goals scored in a season or saves made by a goalkeeper, the features we used were not as objective as we hoped. Some features we used, such as overall rating, are subject to EA (company that developed FIFA) since they determine which attributes are most relevant in determining a player's overall rating.

An issue we had in selecting the best players was that SVM would select the players regardless of what position they played, so sometimes it would give a team with 3 goalkeepers when we only needed 1 goalkeeper. In the future, we could address this by separating the player dataset by position. Strikers, midfielders, defenders, and goalkeepers would be separated and this would allow us to run the model on each position dataset so we can select the appropriate number of players for our ultimate team.