The winner of the next game

Matteo Rusconi, Mohcen Laaroussi

May 1, 2023

Abstract

In this project, a machine learning approach is developed to predict the outcomes of football matches, focusing our work on trying to create features based on analyzing daily news on teams and players, together with statistics regarding the league and seasons of choice. After data collection and models training, a comparison between results shows that text analysis on daily news can be effectively combined with machine learning to accomplish the prediction task.

1 Introduction

Predicting the outcome of soccer matches has been a topic of research for a long time. The complexity of the problem arises from various factors that affect matches outcomes, including the shape of teams and players, and weather conditions. Previous research has shown that combining different sources of data can improve the accuracy of predictions. In this project, we propose a machine learning-based approach that combines text analysis of news articles and the use of statistical data of games and players.

Our approach involves several steps. First, we collected news articles related to the teams playing in the italian Serie A league from the 2017 to 2020 seasons, and performed text analysis on them. Two methods were implemented and compared with each other: We used a pre-trained sentiment analysis model to assign a polarity score to each team based on the sentiment of the sentences that mention the team. The other method is based on calculating a polarity score based on how much each word is related to the victory/loss of a team, computing the point-wise mutual information [1]. We then use these polarity scores as features for our machine learning models.

Second, we collected statistical data related to matches, teams, and players, including goals scored, shots on target, ball possession, and other relevant features. We used this data to further improve the predictive accuracy of our models. Finally we used this data to create new features that add on to the features created on text analysis.

We evaluated the performance of different machine learning models, including decision trees, random forest, KNN, neural networks, XGBoost, ridge regression, and SVM. We compared the accuracy of these models with and without statistical features and we found that neural network and Random forest perform better than the others. We also found that combining text analysis on news and statistical data improves significantly the accuracy of predicting the outcome of soccer matches.

2 Methodology

This section explains how data is acquired from different sources and how it is manipulated in order to construct a machine learning ready dataset that will be used to predict the outcome of football matches. In addition to statistical data regarding teams and players, news articles are also collected and analyzed to extract information that might be useful in the prediction task.

2.1 Data extraction and cleaning

Regarding football news articles, we retrieved them by implementing a python scraper that collected articles about the italian league Serie A from the 2017 to 2020 season, querying the Italian news media

channel Sky Sport [1]. Data about matches was collected using an API provided by "apifootball.com" [2], thus obtaining a total of 1520 matches entries, each of which containing detailed information about the game, such as lineups and in game statistics. For each match day, updated data regarding individual players statistics and teams standings were instead scraped from "understat.com" [3]. Finally, we retrieved some players statistics from the "sofifa.com" database [4], such as generic players overall attributes scores and goalkeepers specific scores.

The extracted data was subjected to data cleaning process to remove matches with missing information and to clean up articles by removing unnecessary words and symbols and lowercasing text.

2.2 Text analysis

News articles were analyzed with the aim to assign scores to each team they are referring to. Scores are numbers that should represent how articles are (positively or negatively) related to matches outcomes.

First of all, we proceeded with splitting articles into sentences and assigning each one to the team it refers to. This is done because an article could be referring to multiple teams. The assignment was done by analyzing each sentence with a natural language processing python library called spacy. The procedure is described as follows:

1. Lemmatize words and remove those that are not nouns, adjectives, adverbs or verbs;
2. Starting from the verbs of the sentence, navigate the left children of the synactic dependency tree to find all possible subjects between the nouns and proper nouns of the sentence with spacy;
3. If a conjunction of two or more teams is present, the sentence gets immediately assigned all of these teams.
4. Otherwise, until a team is assigned to the selected sentence:
   (a) For each found subject:
      i. Check if it corresponds to any team name or team name alias (e.g. Rossoneri for Milan) and assign that team to the sentence
      ii. Check if it corresponds to any player name and assign its corresponding team to the sentence, if there is only one (players may have played for multiple teams in one season)
   (b) If not assigned yet:
      iii. Check if the tags attributes of the article contain references to any team or player name, and assign consequently
      iv. Assign to the last assigned team, since there isn't any new relevant subject it can be assumed that the sentence is still referring to that team.

As explained above, we preferred to look at the subjects of each sentence over blindly using the article tags to detect the referring team. This was done in order to try to achieve maximum precision in associating sentences and teams, which is a key step towards a good prediction. After the procedure each sentence of each article will surely be marked as referring to one or more teams and this is useful to know which team to assign the score.

2.2.1 First approach: TextBlob sentiment analysis

The first of the two methods we implemented is based on assigning a sentiment score to each sentence. We used the "TextBlob"[5] library for processing textual data to do so. Sentiment score indicates the positivity, negativity, or neutrality of the sentence with respect to the team being previously assigned to that sentence. TextBlob utilizes a pre-trained sentiment analysis model that given a word or a sentence outputs a sentiment score ranging from -1 to 1, where 1 is total positivity, -1 is total negativity and 0 is considered neutrality. Of course there can be intermediate values. This simple but

[1] https://sport.sky.it/archivio
[2] https://apifootball.com/
[3] https://understat.com/
[4] https://sofifa.com/players
[5] https://textblob.readthedocs.io/en/dev/

effective approach is already capable of capturing some relevant properties of the articles in terms of describing general football teams trends and shapes. An important aspect that has to be taken into account while using articles to perform a prediction over a specific football match, is the range of days preceding the match on which articles should be considered. In this analysis, we tested ranges of 7, 14 and 21 days. The main obstacle of this procedure was that in general every sentiment analysis tool (including TextBlob) worked significantly better on English language. For this reason, we opted to utilize the translated version of the articles to perform this scoring task. The translation was done using a library that relies on Google Translate.

2.2.2 Second approach: Polarity scores

This second method is inspired by [1]. The main logic behind this approach is to give each word a weight based on how it is correlated to wins, losses or draws. Ideally, if a word appears frequently on articles related to winning teams, it should be seen with a positive polarity, or vice versa.

First, we proceeded to group sentences by related team and by their appertaining article date, in order to construct sets of sentences that can be associated with specific matches. So, looking at every match date, we grouped the sentences in samples of 7, 14 and 21 days preceding the match. Then, every sample is labeled with positive, neutral or negative, based on if the corresponding match was respectively won, drawn, or lost. Summing it up, for every match and team playing in that match, we generated three labeled samples containing sentences found in articles published in the three ranges of days previously specified.

Second, we manually selected a set of keywords ('win', 'lose', 'comeback', 'strong', 'fit', 'recovery', 'lose', 'injured', 'weak', 'disqualified') that we believed to have strong relevance in describing how a team is performing in general and in reflecting people opinions on it and on the next match that team is playing in. Starting from this set of keywords, we proceeded to extend it by computing the Google's Word2Vec vectorization and picking the most similar words (found in the articles) to the manually selected ones. We found new interesting words like 'triumph', 'overtake', 'underdog', 'jeopardize', 'strengthen', 'favorable' and many more.

In order to capture correlations between keywords and matches outcomes, the next step was to compute the so called polarity scores ([2], [3]). To do so, we needed to compute the point-wise mutual information (PMI) given a keyword:

$$PMI(w, pos) = log \frac{freq(w, pos) \times N}{freq(w) \times freq(pos)} \tag{1}$$

where $freq(w, pos)$ denotes the frequency of the keyword w occurring in all positive samples, N denotes the total number of samples, $freq(w)$ denotes the total number of keyword w occurring in all samples and $freq(pos)$ denotes the total number of positive samples. Furthermore, we calculate the polarity score:

$$PS(w) = PMI(w, pos) - PMI(w, neg) + \alpha PMI(w, neut) \tag{2}$$

where α is a scaling factor included to give less importance to draws in polarity scores, and was set to 0.2. We also tried to assign each manually selected keyword a constant bias of ± 0.5 according to if that word is assumed to have a positive or negative polarity (e.g. the keyword "lose" has a -0.5 bias), adding it up in the polarity score formula. Extended keywords inherit biases from their corresponding original keywords (those they've been extended from), scaled by the similarity between the two. Hence, given a sample, its total polarity score is computed as follows:

1. Calculate the polarity score for each keyword in the sample
2. Multiply it by the TF-IDF score of that keyword
3. Compute the sum over all keywords

As a result, after these steps, every match was linked to two polarity scores regarding both home and away teams, computed with respect to articles published within a range of days preceding the match. We empirically evaluated that the 21 days range achieved the best result, both for this approach and for the one explained in the previous subsection.

2.3 Feature engineering

In order to train machine learning models, multiple features were created from the different datasets we collected during the data extraction phase. More specifically, from matches raw data, in-match statistics, such as goals scored/concedes, ball possession percentages and number of shots taken, were extracted and used to calculate a 5-day rolling average of each one, thus having features that represent the average statistic of the 5 previous games. It should be noted that to predict the outcome of a game we used as features only the information about previous games, with the intent to predict the result before that match happens.

To create a feature that summarizes the strength of the teams playing in a match, players ratings were used in conjunction with the lineups playing. The strength of a team is given by the sum of the overall ratings of the players present in the lineup.

Finally, for every match we designed features for the scores obtained during the text analysis phase. Thus, for both methods we created features representing the scores calculated 7, 14 and 21 days preceding the match, with and without the scaling by leaderboard rank and by bias. This resulted in having 18 versions of text scoring features for each match and team pair that will be tested separately in predicting matches in order to choose the best performing one.

2.4 Feature selection

To perform feature selection we used both correlation and importance metrics. We found out that *ppda* and *corner kicks* were the leasts correlated features to the result of matches, with a pearson correlation coefficient smaller than 0.02. On the other hand, as expected, the most correlated features were Fifa *team overall rating*, *leaderboard ranks* and SkySport *articles scores*. Besides these correlations with match results, we found some strong correlations among features like *xGoals* and *xAssists*, or *shots total* and *goal attempts*. Feature importance confirmed correlation results.

As a result, we proceeded to drop *ppda*, *corner kicks*, *xAssists* and *shots total*, thus obtaining a final set of 22 features.

3 Experiments

Once the data extraction, cleaning, feature engineering and selection procedures were completed, ML techniques were applied to this data. We experimented with several models, and compared them with each other to see which one performs the best. These models are:

- Decision Tree
- Random Forest
- XGBoost
- K-Nearest Neighbor
- Neural Network
- Ridge Regression
- Support Vector Machine
- Logistic Regression

3.1 Balancing data

While analyzing the matches, we noticed that the dataset was imbalanced because the number of draws was significantly lower than wins and draws. Therefore, to balance the data, we used a technique called oversampling. Specifically, we duplicated the instances of the minority class (draws) to match the number of instances of the majority class (wins and losses). This allowed us to have a more balanced dataset for training our models.

3.2 Models training

In order to train and evaluate models, we used nested cross-validation. It is a useful technique for model evaluation and selection, especially when dealing with small or imbalanced datasets, and involves using an outer loop and an inner loop of cross-validation to evaluate and tune the hyperparameters of the models.
We used a 5-fold nested cross-validation approach, where the data was randomly partitioned into 5 folds, and 4 folds were used for training, while the remaining fold was used for testing. We repeated this process for each fold of the outer loop.

For each training fold in the outer loop, we performed an inner loop of cross-validation to tune the hyperparameters of the model. Specifically, we used a randomized search approach to search over a range of hyperparameters for each model, and chosen the hyperparameters that resulted in the best performance on the test set. We repeated this process for each fold of the outer loop and calculated the average accuracy across all folds.

Using the nested cross-validation approach allowed us to obtain an unbiased estimate of the performance of our models, as it prevents overfitting by assessing the model's generalization ability on unseen data. The approach also allowed us to choose the best model and hyperparameters with the best accuracy.

3.3 Experimental Results

We first experimented by training the models with news articles scores as the only features, for both of the approaches. We then further trained them adding all other statistical features. The results are represented in the tables below, one for each feature set utilized.
Table 1 summarizes the results while using only *TextBlob* sentiment analysis scores as features. The results show that deep Neural Network had the best performance in terms of accuracy with 51.7%, which is significantly better than chance (33% given the three possible outcomes), while on the other hand, Tree Classifier was the worst performing model with only 43.6% accuracy. As shown in table 2, adding statistical features provides a slight but important boost to the prediction power of our models, allowing Neural Network to reach 57.2% accuracy. Ridge Regression also benefits a lot from the addition of statistics, going from 49.8% to 54.6% of accuracy. Similar performances are obtained by Random Forest model.

	accuracy	precision	recall	f1 score
Tree Classifier	0.436	0.435	0.423	0.418
Random Forest	0.474	0.446	0.443	0.443
XGBoost	0.466	0.426	0.427	0.424
K-NN	0.477	0.456	0.451	0.452
Ridge Regression	0.498	0.433	0.445	0.432
Neural Network	0.517	0.456	0.455	0.439
SVM	0.459	0.461	0.448	0.445
Logistic Regression	0.497	0.452	0.454	0.450

Table 1: TextBlob sentiment analysis

Table 3 shows the performances when polarity score is used as news article scoring feature. We can see that Logistic Regression is the best model and, compared to the previous tested scoring approach, it does achieve a better generalization with 52.1% accuracy. When introducing statistics, Neural Network outperforms every other model with 57.5% accuracy, which is slightly better than the TextBlob approach, therefore becoming the best model in terms of accuracy as shown in table 4.

In general, by observing performances of different models, it is noticeable an increase in accuracy from table 1 to table 3, stating that the polarity score approach is capable of extracting more information from the articles compared to the TextBlob approach.

	accuracy	precision	recall	f1 score
Tree Classifier	0.466	0.448	0.438	0.439
Random Forest	0.546	0.483	0.490	0.481
XGBoost	0.531	0.456	0.467	0.451
K-NN	0.510	0.492	0.483	0.484
Ridge Regression	0.546	0.482	0.490	0.483
Neural Network	0.572	0.519	0.521	0.515
SVM	0.459	0.417	0.417	0.413
Logistic Regression	0.532	0.491	0.489	0.489

Table 2: TextBlob + statistics

	accuracy	precision	recall	f1 score
Tree Classifier	0.461	0.446	0.438	0.437
Random Forest	0.482	0.448	0.446	0.446
XGBoost	0.482	0.434	0.438	0.433
K-NN	0.478	0.445	0.443	0.443
Ridge Regression	0.498	0.431	0.450	0.432
Neural Network	0.516	0.465	0.469	0.465
SVM	0.477	0.464	0.454	0.454
Logistic Regression	0.521	0.460	0.464	0.446

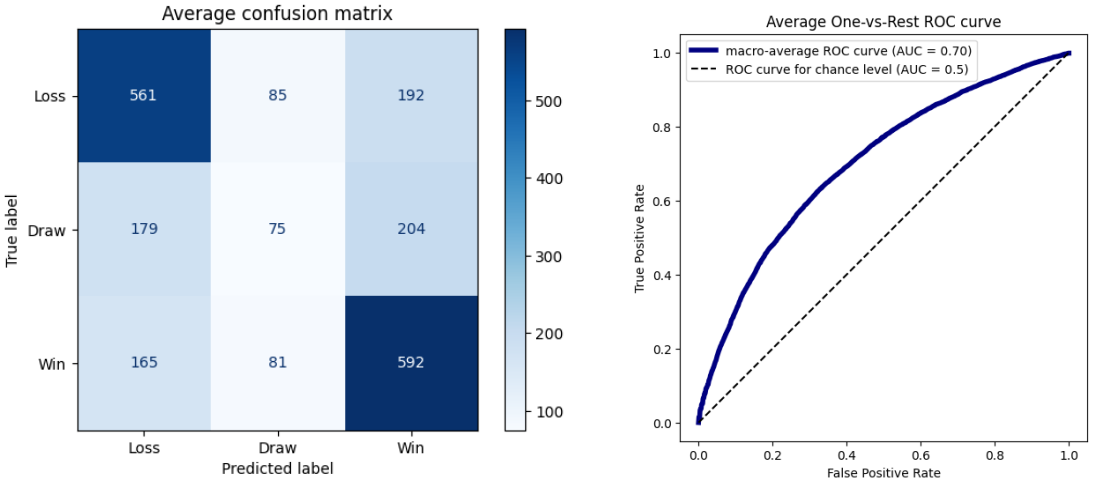
Table 3: Polarity score

	accuracy	precision	recall	f1 score
Tree Classifier	0.476	0.454	0.449	0.448
Random Forest	0.541	0.480	0.486	0.478
XGBoost	0.535	0.458	0.471	0.454
K-NN	0.512	0.493	0.485	0.486
Ridge Regression	0.550	0.482	0.492	0.483
Neural Network	0.575	0.512	0.514	0.502
SVM	0.477	0.428	0.430	0.425
Logistic Regression	0.536	0.490	0.491	0.490

Table 4: Polarity score + statistics

4 Conclusion

In conclusion, we have shown that text analysis can be effectively used to predict outcomes of football matches. By observing the confusion matrix in figure 1, it is clear that correctly classifying draws is really hard compared to classifying wins and losses. This is due to the fact that even if the two teams playing against are approximately on the same level, it's hard to tell that the match will end with a



(a) Averaged confusion matrix

(b) Averaged OvR ROC curve

Figure 1: Neural Network : Polarity scores + statistics performances (best model)

draw. On the other hand, if a top level team plays against a significantly weaker team, it is a lot more likely that it will win, hence predicting wins and losses seem to be a much easier task. A possible extension of this work could be to find some techniques that are more capable of classifying those draw matches or to create more sophisticated features that could potentially help our models to learn more about draws. Furthermore, text analysis on newspaper articles could be improved with new ways of extracting useful information about teams that may help with the prediction task.

References

[1] Yangtuo Peng and Hui Jiang. Leverage financial news to predict stock price movements using word embeddings and deep neural networks. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 374–379, 2016.

[2] Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst*, 21(4):315–346, 2003.

[3] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Artif. Int. Res.*, 37(1):141–188, 2010.