Soccer Match Outcome Prediction

Ali Aljaffer

Department of Computer Science and Engineering
Wright State University, Dayton, OH
Email: aljaffer.3@wright.edu

Abstract. A look into the possibility and viability of predicting the winner of a Soccer match involving two international teams by leveraging machine learning to make accurate predictions based on features involving game performance of the two teams. Logistic Regression (LR), Support Vector Machine (SVM), and Gradient Boosting (GB) were all used as a baseline to make predictions to then be compared to the main chosen model, Random Trees Embedding with Extra Trees Classifier (RTE). The RTE proved better than the baseline models, achieving an accuracy of 0.6453 and an average F1-Score of 0.65. Followed by GB which achieved an accuracy of 0.5445 and an average F1-Score of 0.54. Followed by LR and SVM, which achieved similar results for accuracy 0.4925±0.0005 and an F1-Score average of 0.483±0.003.

Keywords: Soccer predictions, Multi-class Classification, Random Trees Embedding, Random Forest, Feature Extraction, Data Resampling.

1 Introduction

For decades, the popularity of sports has been increasing massively all over the world. One sport, however, is dominating the sports category of entertainment. That sport is Soccer (Or Football, as it's called worldwide.) [1] Due to this popularity, we get to have access to a wide variety of statistics and datasets to provide insight into what makes a team more likely to win a match. Is it the team's recent form? How much influence does the location where the match is played have on the outcome of a game? Is FIFA ranking really that important? We will use machine learning to explore two datasets that can help us get more understanding about the most loved sport in the world by building a predictive model to predict the outcome of a match between two international teams. According to our results, we could leverage our prediction model to our advantage, for instance in sports betting.

2 The Data

For the FIFA rankings, I will use cashncarry's FIFA World Ranking 1992-2023 dataset. [2] And for the match history and outcome, I will use martj42's International football results from 1872 to 2023 dataset. [3]

2.1 FIFA Ranking

A FIFA ranking is a point-based system for international teams. Teams gain points based on their performances in various tournaments played throughout the year. A high-ranking team beating a low-ranking team would earn less points than if the two teams were of similar ranking. This system makes sure that point distribution is fair. FIFA rankings are updated throughout the year.

For the FIFA ranking dataset columns, refer to Table 1.

Table 1. Columns in the FIFA ranking dataset.

Column	Explanation	Data type	Non-null count
rank	FIFA rank	Int64	64757
country_full	Full name of the country	Object	64757
country_abv	Abbreviation of the country name	Object	64757
total_points	Total points so far	Float64	64757
previous_points	Previous number of points	Float64	64757
rank_change	Change in rank	Int64	64757
confederation	Confederation the country belongs to	Object	64757
rank_date	Date that the rank was assigned.	Object	64757

Preprocessing

Only the following columns will be used: rank, country_full, total_points, rank_date. These columns are used to build a Pandas' DataFrame. The DataFrame contains columns that have the rank_date year and rank_date points. The index of the DataFrame is a country column, containing the country name. This way, referencing the name of the country and the year to retrieve the rank of that country at that year is made simpler. Example:

```
rank_at_year.loc['Croatia',2020] # Returns 11
```

Or referencing using the name of the country and year_points to get the FIFA points at that year. Example:

```
rank_at_year.loc['England',2001] # Returns 657
```

This will help later in adding more features to the main dataset, the International Match Results.

2.2 International Match Results, 1872-2023

The dataset contains match results dating back to 1872. The FIFA rankings dataset contains rankings from 1993 and later, so we will discard matches that came before 1993 from this dataset. For the columns, refer to Table 2.

Table 2. Columns in the International Match Results dataset

Column	Explanation	Data type	Non-null count
date	Date the match was played in	Object	45315
home_team	Full name of the home team country	Object	45315
away_team	Full name of the away team country	Object	45315
home_score	Number of goals scored by home team	Int64	45315
away_score	Number of goals scored by away team	Int64	45315
tournament	Tournament the match was played in	Object	45315
city	City the match was played in	Object	45315
country	Country the match was played in	Object	45315
neutral	Whether the match was played in a neutral location	Bool	45315

Preprocessing

For this dataset, the only column we do not need is the City column. We will preserve the remaining columns for data analytics, encoding, and feature preprocessing.

The preprocessing begins by turning the columns with Object data type to an appropriate, numerical data type. The home_team column is encoded to a numerical value that represents the team. To keep this encoded value consistent across all columns that reference the country, the encoded value for each country is stored in a dictionary data structure with keys being the country name, and values being the encoded value. We use this dictionary to encode the away_team and country columns.

The date column is converted from Object type to a datetime type. This is done to enable comparison between two dates, which will help with match filtering and selection.

The tournament column contains many unique values, but the values themselves are not significant, only what they represent. The tournaments are encoded based on what type of tournament they are: 0 for Tournament, 1 for cup, and 2 for friendly. The distinction is needed because: A tournament allows ties, while a cup does not. A friendly game is less serious than either a tournament or a cup game, thus needing its own category.

Finally, the neutral column is converted from bool to int64.

Data splitting

The dataset is split in a 70/30 split. This was found to be an optimal split that maximized the accuracy.

Resampling

Under-sampling was done to balance the data samples given, since there was a larger sample count for class 2 – Home win over the other two classes. See figure 1 for value count. See figure 2 for value counts after resampling.

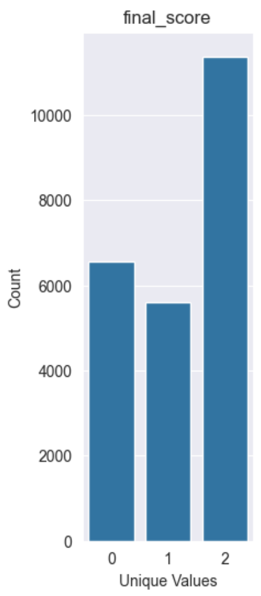

Fig. 1. Value counts of final_score column.

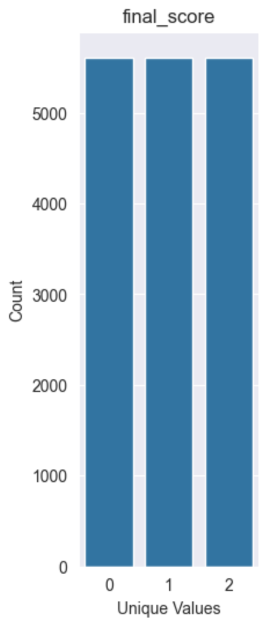

Fig. 2. Value counts of final_score column, post-resample.

Class 0 had 6563 samples, class 1 had 5610, and class 2 had 11375. All were brought down to 5610 samples – matching class 1's count.

Feature addition

Features were extracted from the original dataset and then added as columns to the main dataframe to be used for predictions. Refer to Table 3 for the features that were extracted and added.

Table 3. Features that were engineered and added.

Feature	Engineering method	Data type	Value Range
final_score	Checking if home_score (> or < or ==) away_score,	int64	{0,1,2}
home_code	Encoding the column using the Pandas cat.codes	int64	[0, ∞)
away_code	Using the same encoded value from home_code column	int64	[0, ∞)
hosting_country_code	Using the same encoded value from home_code column	int64	[0, ∞)
tournament_code	If the tournament column contained specific words related to the type of tournament, set to that type.	int64	{0,1,2}
most_recent_rank_(home/away)	Simple lookup of the rank_at_year dataframe with reference to the (home/away) team and the year of the match	int64	[0, ∞)
most_recent_rank_difference	The difference between the value in column most_recent_rank_home and most_recent_rank_away	int64	(-∞, ∞)
home_advantage	Checking if home_code is equal to hosting_country_code	int64	{0,1}
(home/away)_fifa_pts	Referencing the rank_at_year dataframe with the country name and year_points	int64	[0, ∞)
head_to_head_last_5_(home/away)	Creating a match_id column in the format homeTeamName_awayTeamName, then using a rolling window of 5 to get the sum of records that have a value of 2 (in case of home) or 0 (in case of away) in their final_score column	int64	[0,5]
(scored/conceded)_last_5_(home/away)	Grouping by (home/away)_team column and then creating a rolling window of 5 on the (home/away)_score column and getting the sum.	int64	(-∞, ∞)
goal_difference_last_5_(home/away)	Subtracting the scored column and the conceded column that were created previously	int64	(-∞, ∞)
days_since_last_match_(home/away)	Grouping by (home/away)_team columns and getting te number of days prior to the match	int64	[0, ∞)
wins_last_10_games_(home/away)	Checking the final_score column of the last 10 games for 2 values (in case of home) or 0 values (in case of away)	int64	[0,10]

In total, after adding these features, the column count is 23.

3 Methodology

The chosen machine learning model is Random Tree Embedding (RTE). RTE transforms a dataset into a higher-dimensional representation, by coding datapoints according to which leaf of each tree it is sorted into. [4] After transformation, the data is then passed into a classifier model for fitting. The model chosen is Extra Trees

Classifier – A Random-Forest-based model that randomizes its splitting and does not use bootstrapping.

The reasoning for this choice is that our features may not exhibit direct relationships between each other. Transforming them into a higher dimension could help in capturing complex relationships between them, thus leading to better results. Furthermore, the transformation helps with the reduction of underfitting, while also improving the separation of classes for better classification.

Three models will be used as a baseline to compare against the RTE model proposed. These models are: Support Vector Machine (With OneVsOne), Logistic Regression (SoftMax), and Gradient Boosting Classifier.

4 Experimental Results

The results are done on the resampled data (Under-sampled) with no rescaling. The parameter details for the models are as follows:

- SVM
  - decision_function_shape: ovo
  - random_state: 42
  - kernel: rbf
  - C: 1.0
- Logistic Regression
  - multi_class: multinomial
  - solver: lbfgs
  - penalty: l2
  - random_state: 42
  - max_iter: 100_000
- Gradient Boosting Classifier
  - n_estimators: 400
  - random_state: 42
  - loss: log_loss
- make_pipeline for RTE.
  - Random Trees Embedding
    - n_estimators: 400
    - min_samples_leaf:5
    - random_state:42
  - Extra Trees Classifier
    - n_estimators: 400
    - min_samples_split: 3
    - min_samples_leaf:1
    - max_features:log2
    - max_depth: None

4.1 Support Vector Machine (SVM)

SVM performed the worst with an accuracy 0.4928. Refer to Table 4 for metrics of SVM. Refer to Figure 3 for SVM confusion matrix.

Table 4. Support Vector Machine metrics for under-sampled dataset

	Precision	Recall	F1-Score	Support
0	0.53	0.54	0.53	1683
1	0.4	0.35	0.37	1683
2	0.53	0.59	0.56	1683
accuracy			0.49	5049
macro avg	0.49	0.49	0.49	5049
weighted_ avg	0.49	0.49	0.49	5049

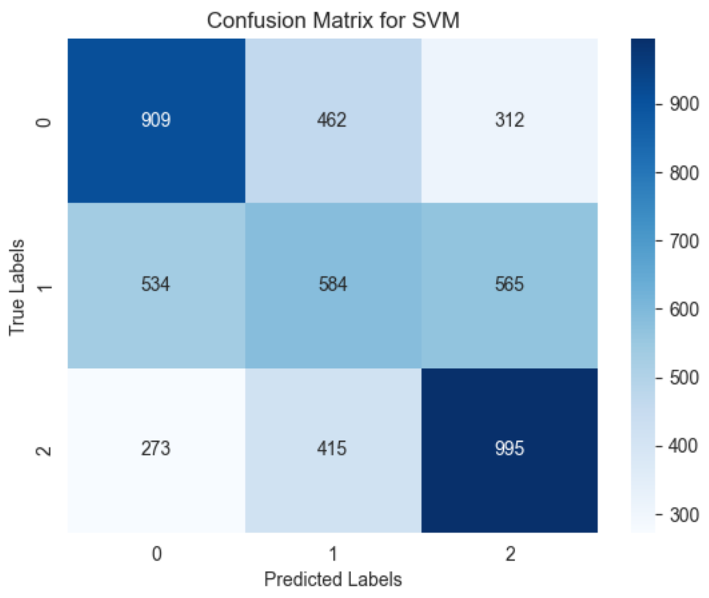

Fig. 3. Confusion Matrix for Support Vector Machine

4.2 Logistic Regression (SoftMax)

Logistic Regression performed close to SVM with an accuracy of 0.4934 but was still bad. See table 5 for metrics. See figure 4 for the confusion matrix.

Table 5. Logistic Regression metrics for under-sampled dataset

	Precision	Recall	F1-Score	Support
0	0.52	0.59	0.55	1683
1	0.39	0.29	0.33	1683
2	0.54	0.6	0.56	1683
accuracy			0.49	5049
macro avg	0.48	0.49	0.48	5049
weighted_ avg	0.48	0.49	0.48	5049

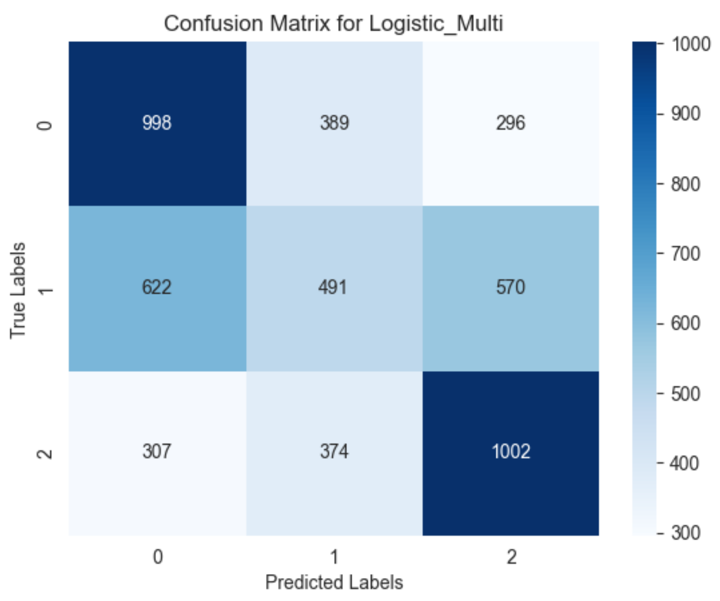

Fig. 4. Confusion Matrix for Logistic Regression

4.3 Gradient Boosting Classifier

Gradient Boosting performed much better than SVM and Logistic Regression. Seems that forest-related models do well with this particular problem. The accuracy was 0.5445. See Table 6 for metrics. See Figure 5 for confusion matrix.

Table 6. Gradient Boosting Classifier metrics for under-sampled dataset.

	Precision	Recall	F1-Score	Support
0	0.56	0.58	0.57	1683
1	0.49	0.48	0.48	1683
2	0.58	0.58	0.58	1683
accuracy			0.54	5049
macro avg	0.54	0.54	0.54	5049
weighted_ avg	0.54	0.54	0.54	5049

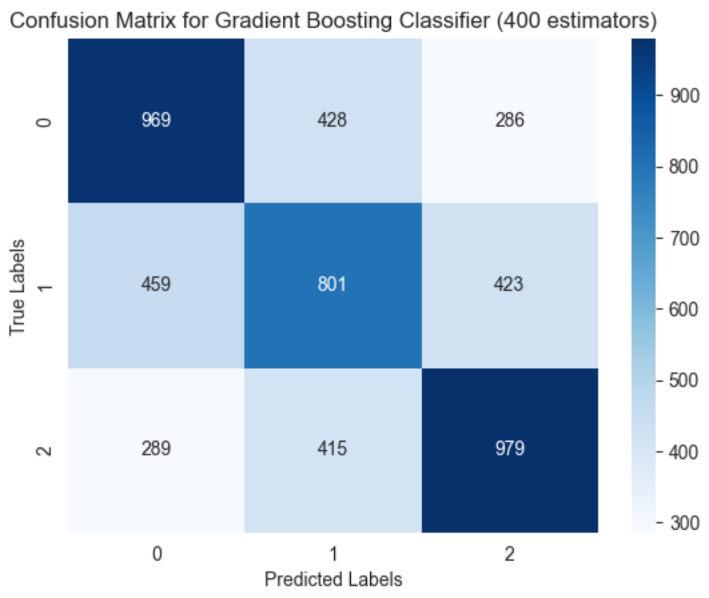

Fig. 5. Confusion Matrix for Gradient Boosting Classifier

4.4 Random Trees Embedding

The best performer out of the models, with an accuracy of 0.6453. See table 7 for metrics. See figure 6 for confusion matrix.

Table 7. Random Trees Embedding with Extra Trees Classifier metrics for under-sampled dataset.

	Precision	Recall	F1-Score	Support
0	0.59	0.67	0.63	1683
1	0.8	0.59	0.68	1683
2	0.6	0.68	0.64	1683
accuracy			0.65	5049
macro avg	0.66	0.65	0.65	5049
weighted_ avg	0.66	0.65	0.65	5049

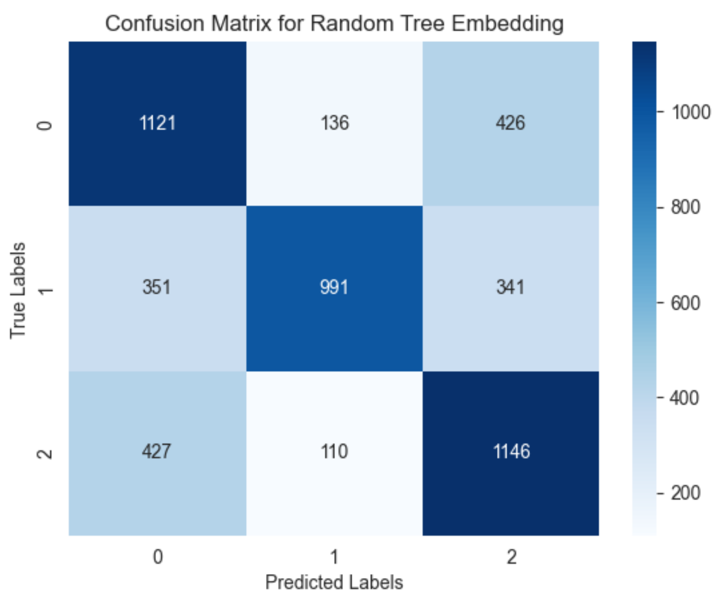

Fig. 6. Confusion Matrix for Random Trees Embedding

5 Discussion

It is apparent that Random Forest-based models performed better for this problem. While SVM and Logistic Regression are great models, they do not seem to be capturing the relationships between features as well as Random Forest does – Or it might just be the randomness of Random Forests that help in capturing these connections. Scaling and oversampling (with SMOTE) did not prove to be beneficial for Random Trees Embedding. See figure 7 for post-SMOTE value counts and table 8 for metrics. It could be the case that the SMOTE operation was done incorrectly because it should, theoretically, improve the results significantly.

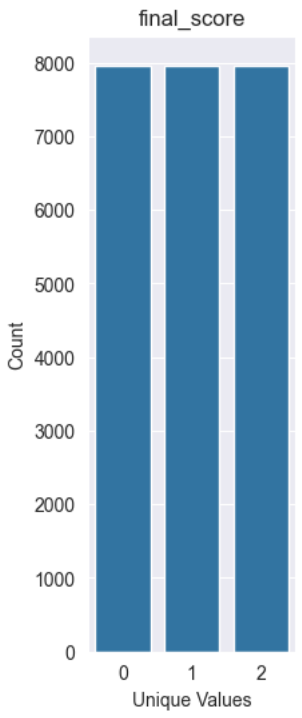

Fig. 7. Value counts of final_score after SMOTE-ing the y_train.

Table 8. Random Trees Embedding with Extra Trees Classifier for SMOTE dataset.

	Precision	Recall	F1-Score	Support
0	0.52	0.50	0.51	1969
1	0.3	0.16	0.21	1683
2	0.62	0.77	0.69	3413
accuracy			0.55	7065
macro avg	0.48	0.48	0.47	7065
weighted_ avg	0.51	0.55	0.52	7065

Further, scaling the features using a Standard Scaler also did not prove beneficial. Unlike the case with SMOTE, where results where worse than without SMOTE, the scaled results are very close to the non-scaled ones. This might indicate that either the model is unaffected by whether the data is scaled or not, or that the data's value ranges are not dissimilar. See Table 9 for results of scaled dataset. The dataset is under-sampled and scaled.

Table 9. Random Trees Embedding with Extra Trees Classifier for Standard-Scaled dataset.

	Precision	Recall	F1-Score	Support
0	0.59	0.66	0.62	1683
1	0.8	0.58	0.68	1683
2	0.6	0.68	0.64	1683
accuracy			0.64	5049
macro avg	0.66	0.64	0.65	5049
weighted_ avg	0.66	0.64	0.65	5049

6 Challenges, Future Steps

Challenges

The dataset used did not contain in-depth statistics that could further help with the model training. Statistics like expected goals (xG), average possession, average number of shots on target, etc. are not provided. These statistics are most likely present in other datasets, in particular ones relating to club matches rather than international matches.

Further, the preprocessing done was tedious and took the majority of the time to develop the final model.

The imbalance of data samples led to low accuracy for the minority class (Tie) which meant there was a need to deal with the imbalance. The chosen strategy, under-sampling, meant that a large portion of the majority class was discarded to bring the amount down to match the minority class. This may have led to losing valuable information relating the features which could lead to better prediction.

Future Steps

An interest arose in Long Short-term Memory after seeing its application and use in other students' projects. It seems like a good candidate for a main model for this problem, as it's a time-sensitive problem that depends more on recent team results than those from months prior, for example. This would definitely lead to better predictions, and would vastly improve the model.

The International Match Results dataset contained different CSV files for penalty shootouts and goal-scorer statistics. These could be leveraged to include more features into the matches dataset for better predictions.

Lastly, a change of problem scope could produce an entirely new problem to solve – game score prediction. Given an input of a match between two teams, predict the final score. This is a multi-output regression problem, which is totally different from our multi-class classification problem at hand. The new problem also has use in betting, as sportsbook keepers also tend to let the customers bet on an exact scoreline between two teams.

7 Conclusion

The metrics for Random Forest-based models proved the superiority of such models over simpler ones like SVM and Logistic Regression. The Random Trees Embedding model ranked first, with great metrics that surpassed every other model by a good margin. Randomization in this model helped reveal better relationships between the presented features, which lead to more accurate predictions. However, there is a need to explore more models that can work on this problem to find the optimal one. Long short-term memory is good candidate for future endeavors with this problem, or any that are of similar interest.

8 References

1. Statistics & Data.: Most Popular Sports in the World, statisticsanddata.org, https://statisticsanddata.org/most-popular-sports-in-the-world/ , last accessed: 2023/12/5.
2. FIFA world ranking 1993-202, cashncarry, Kaggle.com. https://www.kaggle.com/datasets/cashncarry/fifaworldranking
3. International Football results from 1872 to 2023, martj42, Kaggle.com. https://www.kaggle.com/datasets/martj42/international-football-results-from-1872-to-2017
4. RandomTreesEmbedding, scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomTreesEmbedding.html