# Predicting the Outcome of Soccer Matches using Machine Learning

**Project Category: Athletics & Sensing Devices**

**Name: Thomas Brink**
ICME, Stanford University
tbrink@stanford.edu

**Name: Nick Camarena**
ICME, Stanford University
nickcama@stanford.edu

## Abstract

In this report, we compare and evaluate machine learning methods for predicting the outcome of soccer matches, which is a multi-class classification problem suffering from noisy and imbalanced data. To tackle this problem, we employ a variety of machine learning methods in different settings. Our base case suggests that elastic net logistic regression is the best method, reaching an accuracy of close to 51%. Accounting for class imbalance, we find that prediction performance drops slightly. When we increase the dimensionality of our problem, incorporating player-specific data, we find that the different methods have difficulty distinguishing signal from noise, which generally leads to accuracies below 50%. However, after performing dimensionality reduction through truncated SVD, our best method, which is again elastic net logistic regression, obtains an accuracy of 51.6%. Feature importance analysis suggests that away team identifiers for Europe's top teams, such as FC Barcelona and Real Madrid, are most predictive.

## 1 Introduction

Soccer (or: football) is the world's most watched and played sport [1], and the football market is an enormous, multi-billion dollar industry [2]. An extremely popular aspect of soccer is betting on the outcome of matches [3], where people try to earn money by predicting the outcome of soccer matches. Instead of doing so by going with our guts, in this report we investigate the use of machine learning methods to predict the outcome of soccer matches. Since soccer match outcomes are inherently noisy, match outcome prediction is a difficult research problem.

Although the ultimate goal is to develop a more accurate model than that of betting companies, their specific methods and data sources are not publicly available. Fortunately, we can look to public soccer prediction competitions to get a sense of the methods used and a sense for what is possible in terms of accuracy. One noteworthy mention is the 2017 Soccer Prediction Challenge [2]. The winning team used manually constructed features and a gradient boosted tree algorithm [4]. Although this 2017 challenge was scored with a different metric than our classification accuracy, the 3rd place team reported a classification accuracy of 48.7% [5].

The input to our research problem is a database of European soccer matches, from which we encode the outcome as a categorical variable taking on values 'home win', 'draw', or 'away win'. As such, predicting match outcomes is a multi-class classification problem. To tackle this problem, we employ various multi-class classification algorithms which have different advantages and disadvantages. We use decision trees and logistic regression-type methods, which provide a high level of interpretability. In addition, we use the random forest method, an ensemble method which often performs similarly to boosting methods, yet is easier to train and tune [6]. For a final tree-based method, we used XGBoost, which is a highly effective and widely used machine learning method [7]. We also used Support Vector Machines, which can perform well in high dimensional settings [8].

In this paper, we distinguish between four settings of the multi-class classification problem. First, we consider the 'base' case, which includes match, team, and aggregated player data. Second, since

match outcomes are strongly imbalanced, we consider a 'balanced' setting. Third, we perform prediction in a high-dimensional setting by fully incorporating our player data. Last, we project the high-dimensional data onto a lower dimensional subspace using truncated SVD and predict using the lower dimensional data. We find that elastic net logistic regression (ELR) obtains the highest base prediction accuracy, just above 50%, while all methods strongly tend towards predicting the most prevalent class. In our balanced setting, we no longer face this problem, which does, however, come at the cost of accuracy. Similar accuracy loss is found in the high-dimensional case, most likely due to overfitting. In the SVD-reduced case, ELR reaches the highest overall test accuracy at 51.6%. We analyze feature importance and find that away team identifiers for Europe's top teams, such as FC Barcelona and Real Madrid, are most useful for predicting the outcome of soccer matches.

This paper will continue with a discussion of our data, after which we present our methodology and, consequently, provide our results. Last, we give some conclusions and ideas for future work. Our code is available on GitHub.

## 2 Data

In this project, we are working with a soccer database that contains data on various competitions in Europe. More specifically, we have *match*, *team*, *country*, *league*, and *player* data. The database is available on Kaggle [1]. As the format of the database is *sqlite*, we use SQL to process the database into an overarching data table. Since the goal of this research is to predict match outcomes, our level of granularity is that of *matches*. The database contains data on close to 26K matches from 11 European competitions from 2008 to 2016. Every match has certain attributes, such as the outcome, teams involved, players involved, and the date. In order to make use of all relevant information in the database, we link the match table to the other tables based on these attributes. For instance, we have data on teams and their strategies, e.g., what kind of build-up play they use, all of which is time-specific. We make sure that such restrictions are captured when linking different data tables. Due to some matches suffering from a lot of missing data (e.g., team attributes), incorrectly processed attributes (e.g., nonsensical match date), we end up with a dataset containing around 19.5K matches.

For these 19.5K matches, we perform feature engineering in order to create a final data matrix to use for classification purposes. Some variables, such as the stage within a certain season, are numerical, while most features are categorical. For the categorical features, we apply one-hot encoding. We incorporate player features in two ways; our 'base case' (see Section 1) uses aggregated player data, such as the mean and standard deviation of players' age for both teams at the time of a match, as features, while our high-dimensional procedure uses, in addition to the aggregated features, all unique player identifiers (22 per match), again processed using one-hot encoding. Eventually, the base case has around 650 features, while the high-dimensional dataset contains 19.6K features.

Looking at our dataset, we note that we are dealing with an imbalanced class problem. Namely, our output is unevenly distributed across the three match outcomes. Around 46% of matches results in a home win, 29% is an away win, and 25% results in a draw. Since imbalanced data can heavily influence the predictions (e.g., by only predicting a home win, we would reach an accuracy of around 46%), this is an important factor in our methods.

## 3 Methods

### 3.1 Multi-class Classification Methods

**Decision Tree**; The first method we use are decision trees (DT), which split the data in an attempt to separate the different classes. We use cross-validation to select an optimal tree depth between the range of 2 and 50, and then learn a tree of this depth on our training data.

**Random Forest**; Random forests (RF) use an ensemble of decision trees to make classifications, with only a subset of features included in each tree. We use cross-validation to tune both the number of trees and their depth, ranging from 50 to 300 and 2 to 25, respectively.

**XGBoost**; XGBoost uses an ensemble of shallow decision trees, which are combined to improve model performance. In contrast to random forests, XGBoost's decision trees are not trained indepen-

[1] `https://www.kaggle.com/datasets/hugomathien/soccer`

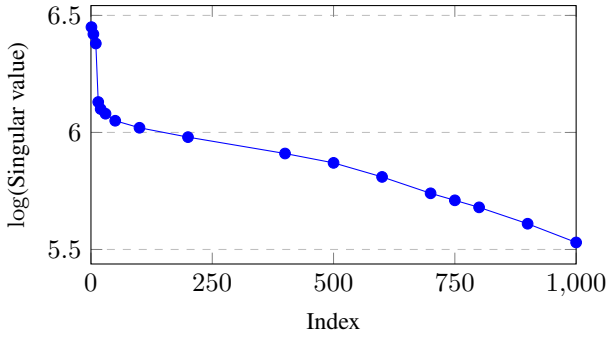


Figure 1: The logarithm of singular values versus their index for the high-dimensional dataset.

dently. That is, the observations we predict incorrectly in a single tree are given more importance in the next tree. Eventually, we combine all predictions made from the different decision trees to classify each observation. As with random forests, we tune XGBoost by choosing the number of trees and the maximum depth of each tree, which we let range from 50 to 200 and 2 to 5, respectively.

**Support Vector Machines**; Support vector machines (SVMs) use a kernel to map the data into a different space in order to separate the classes. We use the 'one-versus-rest' method (instead of the standard 'one-versus-one') to apply SVMs to our 3-class problem. We used cross-validation to tune the regularization parameter $\lambda$ in the range of $\{10^{-i} : -2 \leq i \leq 4\}$.

**Logistic Regression**; Our fifth and last method is logistic regression, which, again using a 'one-versus-rest' scheme, fits a sigmoid function to observations from two classes. With $\ell(\theta)$ being the log-likelihood of our data given the sigmoid function using specific parameters $\theta$, we fit i) $\max\{\ell(\theta)\}$ (plain logistic regression, or LR), ii) $\max\{\ell(\theta) - \lambda||\theta||_2^2\}$ (ridge logistic regression, or RLR), iii) $\max\{\ell(\theta) - \lambda||\theta||_1\}$ (LASSO logistic regression, or LLR), and iv) $\max\{\ell(\theta) - \lambda(\alpha||\theta||_1 + (1-\alpha)||\theta||_2^2)\}$ (Elastic Net logistic regression, or ELR), where we always maximize over $\theta$. For Ridge and LASSO, we have to tune regularization parameter $\lambda$, while Elastic Net requires us to tune both $\lambda$ and $\alpha$. We let $\lambda$ take on values in $\{10^{-i} : -3 \leq i \leq 5\}$ and $\alpha$ range from 0 to 1 with stepsize 0.25. Note that Elastic Net encapsulates LASSO and Ridge as edge cases ($\alpha = 0$ and $\alpha = 1$).

### 3.2 Prediction Settings

We use four settings to test the workings of our methods, all of which use the hyperparameter ranges from Section 3.1. First, we consider the base case, which contains all match and team data, together with aggregated player data. Second, since we are dealing with imbalanced data, i.e., home wins are much more prevalent than other classes (see Section 2), we are expecting that our base case will tend towards predicting home wins. To deal with such imbalance, we adjust class weights inversely proportional to the rate of occurrence of that class. That is, for all observations from class $c \in C$, we set weight $w_c = \frac{n}{|C|n_c}$, where $n$ equals the total number of observations, $|C|$ is the number of classes, and $n_c$ equals the number of observations from class $c$. Our hypothesis is that this method will lead to more balanced predictions, which may, however, lead to lower prediction accuracy.

Third, our high-dimensional setting uses all player information. In this case, the number of dimensions $p > n$, which could lead to overfitting. In addition, due to a high computational burden, we only apply DT, RF, LR, and RLR to this case. DT, RF, and RLR inherently regularize by means of the maximum depth, ensemble trees with feature selection, and the $\ell_2$-norm, respectively. Fourth, to decrease computational costs and the tendency to overfit while retaining most information from the full database, we apply dimensionality reduction to the high-dimensional dataset and fit all methods to the lower-dimensional data. To perform dimensionality reduction, we employ the truncated singular value decomposition, which uses a low-rank approximation that scales well with high-dimensional data. In order to find the dimensionality we should reduce the data to, we look at the singular values of our high-dimensional dataset. We choose to retain 750 dimensions, around which we retain a large portion of variation in the data and use slightly more features than in our base case.

### 3.3 Train-Test Procedure

To evaluate the performance of the different methods, we use a train-validation-test procedure. First, we divide the dataset into train (80%) and test (20%) sets. As some methods are scale-sensitive, we then apply normalization to all features using the train data. Next, we perform K-fold cross-validation ($K = 4$) on the train set using the hyperparameters from Section 3.1. In doing so, we maximize prediction accuracy. After having selected the parameter settings leading to the highest validation accuracy, we refit our model using the entire train set (80%) and evaluate on the test set.

The main evaluation criterion we use is the prediction accuracy (*Acc*), which computes the fraction of predictions that are correct. We choose this metric as our main (validation) metric since our eventual goal is to correctly predict as many matches as possible. In addition, we provide $F_1$-scores, which uses class-specific precision and recall to form an aggregated measure. Furthermore, to assess the class-specific prediction performance, we assess confusion matrices.

## 4 Results

As discussed in Section 3, we consider four prediction settings. We provide accuracies and $F_1$-scores for all methods in all four settings in Table 1. For the base setting, we find that elastic net logistic regression (ELR), where the optimal $\lambda$ and $\alpha$ are 10 and 0.5, respectively, performs best in terms of accuracy, reaching a value of near 51%. In terms of $F_1$, plain logistic regression (LR) performs best, reaching a value of 0.414. This indicates that LR does best in predicting match outcomes while retaining relatively strong precision and recall for the different classes. As it appears, this is the case for all four different settings.

As for the balanced setting, we find that most methods perform significantly worse than in the base case, reaching accuracies close to or below the fraction of home wins in the dataset (46%). The only exception is LLR ($\lambda = 100$), which reaches an accuracy of 50.3% but a lower $F_1$-score than in the base case. Thus, LASSO still tends to favor the most prevalent class. As expected, all other methods have higher $F_1$-scores than in the base case, as we now focus on high precision and recall for all classes. However, our eventual goal is to get as high of an accuracy as possible. Thus, our models prefer the original, imbalanced data.

Looking at the high-dimensional case, where we fit four different methods, we find that all accuracies and $F_1$-scores tend to be lower or similar to the base case, which is indicative of our methods tending to overfit. The best method now is ridge logistic regression (RLR) with $\lambda = 10^5$, but its performance is more than a percent lower than in the base case. All in all, incorporating all player-specific data does not seem to be beneficial. In fact, we would do better without. For the last, dimensionality-reduced, setting, the best-performing model is again elastic net logistic regression (ELR) with $\lambda = 1$ and $\alpha = 0$, which now reaches an accuracy of 51.6%. This is really quite a high number, as the most advanced and best-performing method known for this dataset reaches around 53% accuracy. Overall, the SVD-reduced setting seems to improve model performance in terms of accuracy, while $F_1$-scores remain relatively constant compared to the base case.

Table 1: *Accuracy (Acc) and $F_1$-score for all multi-class classification methods in the four different settings. The best performing method per setting is bolded.*

| | Base | | Balanced | | High-dimensional | | SVD-reduced | |
|---|---|---|---|---|---|---|---|---|
| Method | *Acc* | $F_1$ | *Acc* | $F_1$ | *Acc* | $F_1$ | *Acc* | $F_1$ |
| DT | 0.470 | 0.285 | 0.397 | 0.373 | 0.470 | 0.286 | 0.477 | 0.312 |
| RF | 0.488 | 0.322 | 0.464 | 0.395 | 0.478 | 0.273 | 0.490 | 0.314 |
| XGBoost | 0.500 | 0.359 | 0.433 | 0.415 | | | 0.498 | 0.368 |
| SVM | 0.498 | 0.390 | 0.452 | 0.434 | | | 0.505 | 0.387 |
| LR | 0.502 | **0.414** | 0.459 | **0.440** | 0.401 | **0.375** | 0.501 | **0.411** |
| RLR | 0.507 | 0.402 | 0.461 | 0.439 | **0.496** | 0.325 | 0.511 | 0.402 |
| LLR | 0.501 | 0.398 | **0.503** | 0.384 | | | 0.514 | 0.389 |
| ELR | **0.509** | 0.380 | 0.461 | 0.439 | | | **0.516** | 0.400 |

Before, we saw that ELR is the best-performing method in general, reaching the highest overall accuracy in the SVD-reduced setting. However, in terms of $F_1$-score, ELR performs best in the balanced setting. To see what causes this difference, Table 2 shows confusion matrices for ELR in the SVD-reduced and balanced settings. From the SVD-reduced setting, we find that ELR mainly predicts home wins and only barely predicts a draw. This does not make for balanced predictions. In the balanced setting, we find that ELR now much more often predicts draws and away wins. However, the number of correct predictions (looking at the diagonal entries) is lower than in the SVD-reduced case, mainly since the accuracy for predicted draws is extremely low.

Table 2: *Confusion matrices for ELR in the SVD-reduced setting (left) and ELR in the balanced setting (right).*

| SVD-reduced | **Prediction** | | |
|---|---|---|---|
| **True value** | Home | Draw | Away |
| Home win | 1472 | 65 | 222 |
| Draw | 671 | 56 | 244 |
| Away win | 608 | 64 | 468 |

| Balanced | **Prediction** | | |
|---|---|---|---|
| **True value** | Home | Draw | Away |
| Home win | 912 | 446 | 401 |
| Draw | 323 | 307 | 341 |
| Away win | 252 | 327 | 561 |

We saw that the models that lead to the best predictions are base-case ELR and SVD-reduced ELR, where the latter setting leads to slightly higher accuracy. To interpret the models formed in these settings and assess feature importance, we plot the (standardized) magnitude of feature coefficients for both settings in Figure 3. Note that, since we normalize our data before running our models, the magnitude of the coefficients can be used as a measure for feature importance. From this figure, we find that, as expected, the first 10-20 features have extremely high coefficients compared to the rest of the features. This effect is strongest in the SVD-reduced case, which makes sense, as the first components in truncated SVD explain most variation in the data, whereas later coefficients have much less explanatory power. For both settings, the feature importances move towards 0.

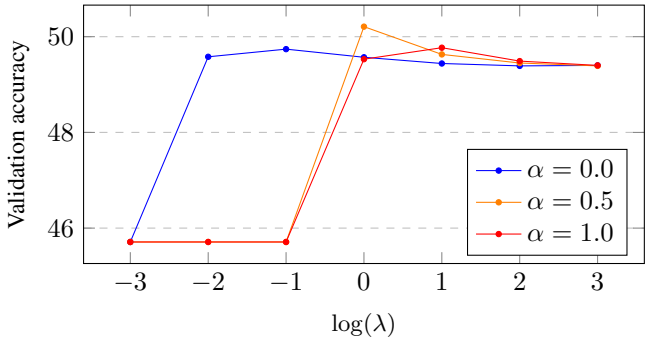


Figure 2: Validation accuracy for ELR for different hyperparameter settings.

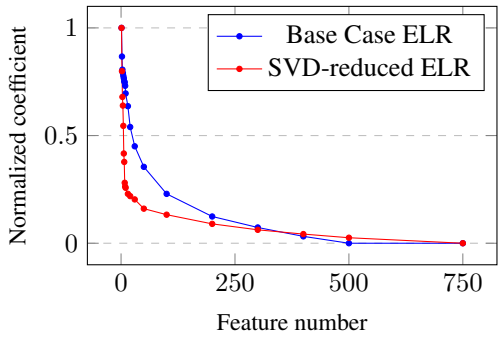


Figure 3: Feature importances for ELR in the base and SVD-reduced case.

Now, an ideal method could zoom in on the top features in Figure 3 and state, qualitatively, which data attributes are of most predictive value. In the SVD-reduced case, our features are now linear combinations of data attributes, such that the features from Figure 3 are not easily interpretable. For the base case, however, we do know what each feature represents. Looking at our top five features in the base case, we find that the most significant features are: i) *Away team = Barcelona*, ii) *Away team = Real Madrid*, iii) *Away team = Bayern Munich*, iv) *Away team = Benfica*, and v) *Home team = Benfica*. We can immediately recognize these teams as powerhouses in European soccer that tend to win the vast majority of their games. It also makes sense that knowing a good team is playing an away game is more important than knowing they are playing at home, because our model will already have a tendency to pick the home team, i.e., home teams are overall more likely to win already.

## 5 Conclusion and Future Work

In this report, we explore multi-class classification methods for predicting the outcome of soccer matches. We do so in four settings, all of which test distinct performance features of our methods. We find that elastic net logistic regression with SVD-reduced data performs best overall, reaching an accuracy of 51.6%. It seems that, by tuning the degree of regularization, logistic regression best separates the signal from the noise. When we account for balanced or high-dimensional data, logistic regression-type methods still perform best, but accuracies are lower than in the base and SVD-reduced settings. Zooming in on feature importance, we find that away team identifiers for Europe's elite are most important. More specifically, FC Barcelona playing away is the top predictor.

For future work, there are several directions we could take. First, we could use our methods to design betting strategies; e.g., we could output probabilistic predictions for each outcome of a match and use these probabilities to design a strategy (we might want to bet on FC Barcelona playing an away game). Second, we could focus more on feature engineering. Gathering and designing features could be an entire project by itself, since any kind of (temporal) features could be designed or scraped, e.g., think of player ratings from the FIFA video game. Third, we could explore the use of hierarchical classification methods for predicting the outcome of soccer matches. This type of method divides the prediction problem in different steps, which can lead to higher accuracy and more balanced outcomes. We have provided code for this method on GitHub but could not incorporate the method in this report yet due to computational resources.

## 6 Contributions

For this project, both of us contributed equally to the outcome. Thomas worked on data cleaning and preparation using SQL, while Nick performed feature engineering on the cleaned dataset. We collaborated on discussing which methods to use and implementing them. Also, we contributed equally to processing and formatting the results, writing our report, and creating a poster presentation.

## References


[1] Ellis Cashmore and Kevin Dixon. *Studying football*. Routledge London, 2016.

[2] Daniel Berrar, Philippe Lopes, Jesse Davis, and Werner Dubitzky. Guest editorial: special issue on machine learning for soccer. *Machine Learning*, 108(1):1–7, 2019.

[3] Hibai Lopez-Gonzalez and Christopher D Tulloch. Enhancing media sport consumption: Online gambling in european football. *Media International Australia*, 155(1):130–139, 2015.

[4] Ondrej Hubacek, Gustav Sourek, and Filip Zelezny. Learning to predict soccer results from relational data with gradient boosted trees. *Machine Learning*, 108(1):1–7, 2019.

[5] Alkeos Tsokos, Santhosh Narayanan, Ioannis Kosmidis, Gianluca Baio, Mihai Cucuringu, Gavin Whitaker, and Franz Király. Modeling outcomes of soccer matches. *Machine Learning*, 108(1):77–95, 2019.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Random forests. In *The elements of statistical learning*, pages 587–604. Springer, 2009.

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[8] Derek A. Pisner and David M. Schnyer. Chapter 6 - support vector machine. In Andrea Mechelli and Sandra Vieira, editors, *Machine Learning*, pages 101–121. Academic Press, 2020.