iJournals: International Journal of Software & Hardware Research in Engineering
ISSN:2347-4890, ijournals.in/ijshre
Volume 12 Issue 10, October 2024

Using Machine Learning to Forecast Football Shot Outcomes

Author: Udit Mishra

Affiliation: Jayshree Periwal International School

E-mail: udit57mishra@gmail.com

DOI: 10.26821/IJSHRE.12.10.2024.121002

ABSTRACT

Machine learning algorithms are employed on a football data set to forecast whether a shot results in a goal. Results from existing models are improved upon by employing various additional algorithms. At the initial stage, the study gathered data from an existing Kaggle article published online by Usama Waheed. The data set includes data from various leagues and their matches. The research analysed shot events, filtered out non-relevant events, and created features based on shot coordinates, angles, player skill, and shot type. Nine machine learning algorithms were implemented: logistic regression, XGBoost, random forests, support vector machines, k-nearest neighbours, decision trees, LightGBM, CatBoost, and artificial neural networks. The research also focused on removing the redundancy, optimizing performance on imbalanced datasets, and fine-tuning model hyperparameters by employing nested cross-validation. Lastly, the models were evaluated on accuracy, precision, recall, and training time metrics. The results revealed insights into the strengths and weaknesses of each model in predicting goals, with specific emphasis on areas of improvement for shot-based football analytics. Decision trees produced the most accurate predictions on the test set.

Keywords: Football, goals, machine learning, prediction.

1. INTRODUCTION

The swish sound of the ball going inside the basket, the crack of the racket hitting the ball, and the splash of divers diving into the pool are extremely soothing. As a national squash player, sports have always been at the top of my list. Confronting the complex and magical ability of supervised learning algorithms has always surprised me [1]. During a recent football match, I learned about several algorithms that specialize in predicting match outcomes by incorporating various factors. The one that intrigued me the most was XGBoost.

In recent years, football prediction models have become increasingly effective and recognised due to their ability to predict matches accurately. Models like XGBoost play a crucial role in allowing various football teams and coaches to make their teams stronger and base their strategy according to the model's predictions. Moreover, these algorithms have also become very well-known in the betting industry as people who are unaware of the sport use them to increase their chances of winning a bet [2]. Research in this field has advanced drastically as researchers are using various machine learning algorithms to understand patterns and base their predictions on them. Hence, these models have earned their place in the field of game prediction, shot prediction, and the number of goals a person can make in a match [3,4,5].

As a passionate athlete, I have always been fascinated by how data intersects with performance on the field. This led me to explore how XGBoost and other machine learning algorithms could be applied to enhance decision-making in football. The ability to predict shot outcomes based on factors such as player rank, position on the field, and shot type is a powerful tool for teams and coaches. It provides insights that can improve strategy and player performance.

By delving deeper into machine learning models like XGBoost, I aim to contribute to the growing field of sports analytics. Combining my passion for sports with a curiosity for algorithmic learning, I seek to apply these tools to improve predictive models in not only football, but also a wide range of sports. This pursuit blends my athletic experience with a

desire to understand and optimize performance through data-driven methods.

To understand more about this topic, I decided to read several papers.

1.1 Literature review

The article "Predicting Football Results Using Python and the Dixon and Coles Model" discusses an extended version of a model built around the Poisson distribution (a distribution that is used to model the number of times an event happens in a fixed interval of time or space) named "Dixon and Coles" [6]. It introduces a problem from a previous discussion, where the model struggled to predict lower scorelines. Hence, it showcases the need to employ Dixon and Coles. This model extends the traditional Poisson distribution to predict football match scores, adjusting for the correlation between the number of goals scored by each team. Lastly, the author introduces a term named "time decay" and its optimization, which is mainly used to increase accuracy by decreasing the influence of older matches, focusing more on recent performance.

Moreover, the second article, "Match Outcome Prediction in Football", starts by introducing the primary goal of the model, which is to predict match outcomes better than bookkeepers [7]. The article from Kaggle delves into various machine learning techniques to predict the outcomes of football matches. The author discusses the importance of features such as team strength, recent performance, home advantage, and historical data. The study implements several models, including logistic regression, decision trees, and ensemble methods like random forests and gradient boosting. The author emphasizes feature engineering and selection as critical steps in improving model accuracy, demonstrating that machine learning can effectively predict match outcomes by capturing complex patterns in the data.

In the article "Football Match Prediction" from Kaggle, the author explores the application of machine learning models to predict the outcomes of football matches [8]. The study uses a dataset containing various features like team form, head-to-head records, and player statistics. Models such as support vector machines (SVM), *k*-nearest neighbours (KNN), and neural networks are implemented to assess their predictive capabilities. The author highlights the importance of data preprocessing and feature selection, showing that incorporating diverse features can enhance the prediction accuracy of the models. The study concludes that advanced machine learning techniques can offer significant improvements over traditional statistical methods in football match prediction.

The article "Expected Goals & Player Analysis", also from Kaggle, provides an in-depth analysis of the expected goals (xG) metric, which estimates the likelihood of a shot resulting in a goal [9]. The author explains how xG models are built using various features such as shot location, shot type, and assist type. The study also explores player performance analysis, using xG to evaluate players' finishing abilities and decision-making in front of the goal. By comparing actual goals to expected goals, the author identifies over- and under-performers, offering valuable insights for player scouting and tactical adjustments.

2. METHODOLOGY

The overview of the methodology includes several steps:

2.1 Primary research

Firstly, I read several research papers and understood the concept of XGBoost and other machine learning algorithms, which piqued my interest.

2.2 Understanding the Article

Then, I picked the article that fascinated me the most. Specifically, the article "Expected Goals (xG) Model" by Usama Waheed, published in May 2023, taught me the workings and mechanics of various algorithms [10]. The original program involved essential feature extraction, event filtering (like shot events), and model-building techniques such as logistic regression for expected goal prediction. A ranking system was also established to provide a better prediction model when calculating the quality of shots of the players.

2.3 Data collection

Next, a sizable data set of teams, players, shots, and player ranks was needed to start the project. This was collected from Waheed's article [10]. This dataset included comprehensive event data from multiple football leagues, focusing on shot events, player rankings, and match details. I used the author's approach as a base to further develop the model.

2.4 Additions and explanations of the features

My code was mainly based on the original code. However, I made some minor changes. For example, the features engineered in the first version included:

a. *x*, *y* coordinates of the shot
b. Shot angle relative to the goal
c. Foot used for the shot (left or right)
d. Position of the shot (low, centre, or high)
e. Match half (first or second)

I kept all these features and incorporated them into my code; however, there were some minor additions to my model. My model included features like:

a. transformations to better capture the relationship between shot distance, angle, and goal-scoring likelihood; and
b. new categorical features for shot outcomes.

2.5 Addition of new machine learning algorithms

The main goal of the research was to introduce new machine learning algorithms to improve forecasting.

The first version of the code in the article included the following algorithms:

a. Logistic Regression
b. XGBoost (Extreme Gradient Boosting)
c. Random Forest
d. Support Vector Machine (SVM)

I introduced new machine learning algorithms. These are the following algorithms I included:

a. Decision Trees
b. LightGBM
c. CatBoost
d. Artificial Neural Networks (ANN)

2.6 Multicollinearity and Feature Selection

To make the model more robust, I performed a Variance Inflation Factor (VIF) analysis, which allowed me to eliminate multicollinear features. Some features, such as foot used and match period, exhibited multicollinearity in both the original and new versions of the model. Furthermore, I removed any redundant features that would have slowed the model down.

2.7 Cross-Validation and Model Selection

Both models utilized stratified *k*-fold cross-validation to balance class representation across training and testing datasets. However, I used nested cross-validation for hyperparameters incorporated in algorithms like XGBoost and LightGBM.

2.8 Final Model Performance

After adding the new algorithms and all the additional features to make the code faster, I ran the model to measure precision, recall, and accuracy on both training and testing datasets, comparing performance across the models.

2.9 Visualization

Lastly, after running the models, I had trouble understanding the results, so I added several graphical representations using the original code from Kaggle. The original code helped me create graphs of shot distributions, goal probabilities from different areas on the pitch, and the importance of features

2.10 Hyperparameter Tuning

Each machine learning model underwent hyperparameter tuning using grid search within the inner loop of the nested cross-validation process. The models were optimized based on F_1-score, a critical metric for imbalanced datasets, to ensure accurate predictions of goals.

2.11 Model Evaluation

The performance of each model was evaluated using a confusion matrix, along with precision, recall, and accuracy metrics for both the training and testing sets. Additionally, feature importance plots were generated for tree-based models like Random Forest, XGBoost, and LightGBM to understand the most influential features for predicting goals.

2.12 Model Comparison

The performance of all the models was summarized in a comparative table showing their training and testing precision, recall, accuracy, and training time. The models were ranked based on their accuracy, precision, recall, and F_1-scores to determine the best-performing algorithm for predicting shot outcomes in football matches.

Following this methodology allowed me to successfully outperform the existing algorithms in the Kaggle code and create a more precise and accurate model.

2.13 Code

The code is available here: https://github.com/udittmishra/USING-MACHINE-LEARNING-TO-FORECAST-FOOTBALL-SHOT-OUTCOMES-CODE

3. RESULTS AND DISCUSSION

Table 1 shows the performance of the algorithms, including runtimes.

Table 1. Performance of the algorithms

Model	Training Time (mins)	Training Accuracy	Training Precision	Training Recall	Testing Accuracy	Testing Precision	Testing Recall
Logistic Regression	0.3243	0.8285	0.3723	0.9538	0.8206	0.3434	0.9533
XGBoost	5.5803	0.9213	0.5686	0.9970	0.8794	0.4321	0.8170
Random Forests	7.1542	0.9371	0.6282	0.9636	0.8954	0.4710	0.7371
SVM	40.8759	0.8451	0.3989	0.9742	0.8281	0.3489	0.9152
k-nearest Neighbours	0.8008	0.9455	0.7772	0.6650	0.9095	0.5311	0.4828
Decision Trees	9.6607	0.9260	0.7052	0.4922	0.9274	0.6566	0.5098
LightGBM	0.0480	0.9113	0.6255	0.3588	0.9159	0.5992	0.3710
CatBoost	3.6423	0.9325	0.7320	0.5506	0.9225	0.6144	0.5147
ANN	18.8320	0.9172	0.6994	0.3535	0.9239	0.6765	0.3956

3.1 Analysis of Table 1

3.1.1 Training Time

- **SVM** has the longest training time at 40.8759 minutes, followed by ANN with 18.8320 minutes.
- **Logistic Regression** is the second most time-efficient model, with a training time of 0.3243 minutes.
- **LightGBM** is highly efficient, completing training in just 0.0480 minutes, making it faster than Logistic Regression.

3.1.2 Training Accuracy and Precision

- ***k*-Nearest Neighbours (KNN)** has the highest training accuracy of 0.9455, followed by Random Forests with 0.9371.
- **Decision Trees Classifier** also has high training accuracy (0.9260), but its precision (0.7052) is lower, indicating possible overfitting.
- **XGBoost** provides a balance with a training accuracy of 0.9213 and training precision of 0.5686.
- **CatBoost** demonstrates strong performance with a training accuracy of

0.9325 and the second-highest training precision of 0.7320 among all models.

3.1.3 Testing Accuracy and Precision

- **Decision Trees** achieve the highest testing accuracy at 0.9274, followed by ANN at 0.9239.
- ***k*-Nearest Neighbours (KNN)** performs well with testing accuracy of 0.9095, though its precision is lower at 0.5311.
- **XGBoost** has a testing accuracy of 0.8794 and a precision of 0.4321.
- **ANN** leads in testing precision with 0.6765, making it highly effective in distinguishing between true positives and false positives.

3.1.4 Recall

- **XGBoost** has strong recall for both training (0.9970) and testing (0.8170), indicating its effectiveness in capturing true positive outcomes.
- **Logistic Regression** has a high testing recall of 0.9533, though its lower precision means it is more prone to false positives.

3.1.5 Performance Across Models

- **Random Forest** is a strong all-around performer, with the highest average accuracy, precision, and recall across both training and testing datasets (0.7721).
- **XGBoost** generalizes well, and has the highest average accuracy, precision, and recall across the testing dataset (0.7095).
- **CatBoost** and **k-Nearest Neighbours (KNN)** had the highest average accuracy over the training and testing datasets (0.9275).

3.2 Improvements in Key Algorithms

To make the model more robust, I performed a Variance Inflation Factor (VIF) analysis, which allowed me to eliminate multicollinear features. Some features, such as foot used and match period, exhibited multicollinearity in both the original and new versions of the model. Furthermore, I removed any redundant features that would have slowed the model down.

3.3 Analysis of Algorithms' Performance

3.3.1 XGBoost

- **Old Version**: The previous XGBoost model had decent performance but took longer to train than simpler models like Logistic Regression.
- **New Version**: In the updated version, XGBoost maintains the same testing accuracy (0.8794) and precision (0.4321) as before, indicating stable performance in identifying true positives and minimizing false positives. However, the most significant improvement lies in the drastic reduction in training time (5.5803 minutes), making the algorithm much more efficient without compromising accuracy or precision.
- **Beats the Old Algorithms**: Although the testing metrics for accuracy and precision remain unchanged, the updated XGBoost model stands out due to its optimized training process, significantly decreasing the time required to train the model. This improvement makes it a more practical choice for larger datasets, as it now delivers the same level of performance with far greater efficiency, closely competing with Random Forests in terms of both accuracy and speed.

3.3.2 CatBoost

- **Old Version:** This model was not present in the original code but is now one of the standout additions.
- **New Version:** CatBoost has emerged as a competitive option, with 0.9325 training accuracy and 0.9225 testing accuracy. Its testing precision (0.6144) is also notable, indicating a very high capacity for distinguishing between goal and non-goal events. This model is fast (training time: 3.64 minutes), making it a good choice for large datasets.
- **Beats the Old Version:** Compared to older algorithms like Logistic Regression, CatBoost significantly improves accuracy and precision. It offers a much more efficient performance while handling complex data.

3.3.3 LightGBM

- **Old Version:** Like CatBoost, LightGBM was not previously included in the original code.

- **New Version:** LightGBM has shown solid results, with 0.9113 training accuracy and 0.9159 testing accuracy. However, its precision (0.5992) on testing data shows that while it captures true positives effectively, it sometimes misclassifies non-goal events as goals. The training time of just 0.048 minutes makes it an extremely efficient model.
- **Beats the Old Algorithms:** LightGBM outperforms earlier models like Logistic Regression, KNN, and SVM in terms of training time and testing accuracy. It is a faster, more lightweight option that sacrifices only a bit of precision for its speed.

3.3.4 Artificial Neural Networks (ANN)

- **Old Version:** ANNs were absent in the first program, and this updated version introduced them as a potential model.
- **New Version:** The ANN achieves 0.9172 training accuracy and 0.9239 testing accuracy. However, its recall (0.3956) on testing data is relatively low compared to the tree-based models, suggesting some inefficiency in distinguishing between close events. The 18.83-minute training time makes it the second slowest of all models, a trade-off for the complexity of neural networks.
- **Beats the Old Algorithms:** Although ANN does not outperform models like XGBoost or Random Forests in recall, it offers a solid alternative with high precision. Its inclusion highlights the growing range of algorithmic choices for improving xG models.

3.3.5 Decision Trees

- **Old Version**: Decision Trees were absent in the first code and are introduced in this updated version as a new model.
- **New Version:** The Decision Tree model achieves a training accuracy of 0.9260 and a testing accuracy of 0.9274, indicating its ability to generalize well from training to unseen data. However, its testing recall (0.5098) suggests that while it performs well, it may miss some positive cases, particularly when compared to models like Random Forests.
- **Beats the Old Algorithms:** Even though Decision Trees are newly introduced, they offer competitive results, particularly in precision (0.6566). The model also benefits from a reasonable training time of 9.66 minutes, making it a viable option for enhancing xG models with quick and accurate predictions.

3.4 Comparison to Older Algorithms

3.4.1 Logistic Regression

Although Logistic Regression is quick to train, with the second shortest training time of 0.3243 minutes, it ranks lower in both testing accuracy (0.8206) and precision (0.3434). While still efficient for simpler tasks, it is clearly outclassed by newer algorithms like XGBoost, CatBoost, and Random Forests, which handle more complex shot event data more effectively. Logistic Regression's inability to capture this complexity results in poorer predictive performance compared to the more advanced models.

3.4.2 SVM and k-Nearest Neighbours (KNN)

SVM takes a long time to train (40.8759 minutes) and, despite this, does not match the performance of faster models like XGBoost or CatBoost. With a testing accuracy of 0.8281, SVM lags behind the newer, more optimized models.

***k*-Nearest Neighbours (KNN)** has improved with strong testing accuracy (0.9095) but still falls behind in precision (0.5311) and recall (0.4828) compared to algorithms like Decision Trees and CatBoost, which provide better overall performance.

4. CONCLUSION

In conclusion, this research has been a resounding success, for several reasons. The updated code has significantly enhanced the performance of the expected goals (xG) model by integrating several advanced algorithms. The improvements in the XGBoost model, along with the introduction of powerful new algorithms such as CatBoost and LightGBM, have dramatically increased the accuracy and precision of the model, surpassing the traditional models like Logistic Regression and *k*-Nearest Neighbours.

Furthermore, introducing tree-based models, such as random forest and gradient boosting algorithms like XGBoost and LightGBM, has helped the model better balance training time, accuracy, and recall values. While artificial neural networks (ANN) demonstrate potential with high precision, their long training time makes them less practical compared to the more efficient algorithms like CatBoost and XGBoost. These enhancements underscore the

importance of selecting the right model architecture for specific datasets, particularly in sports analytics, where precision and speed are crucial for real-time analysis.

The new algorithms outperform the old algorithms in performance and show a clear path forward in advancing expected goals (xG) modelling, paving the way for faster, more accurate, and robust predictions in football analytics. This demonstrates the success of incorporating advanced machine learning algorithms into traditional sports modelling frameworks.

5. ACKNOWLEDGMENTS

Dr. Martin Sewell helped throughout the research paper and allowed me to learn about classification models, etc.

6. REFERENCES

[1]. GeeksforGeeks, “Supervised vs Unsupervised Learning,” GeeksforGeeks, September 2024.

[2]. A. J. Owen, M. F. James, and S. J. Dolan, “Forecasting and decision analysis in football betting,” International Journal of Forecasting, vol. 21, no. 2, pp. 331-340, Apr.-Jun. 2005.

[3]. R. Baker and I. McHale, “Forecasting exact scores in National Football League games,” International Journal of Forecasting, vol. 29, no. 1, pp. 122-130, 2013.

[4]. R. Mattera, “Forecasting binary outcomes in soccer,” Annals of Operations Research, vol. 325, pp. 115-134, Aug. 2021.

[5]. “Predicting outcomes in football: A statistical analysis,” Journal of Sports Analytics, vol. 7, pp. 77-97, 2021.

[6]. Martin Eastwood, “Predicting Football Results Using Python and the Dixon and Coles Model”, Pena.lt/y, June 2021.

[7]. Airback, “Match Outcome Prediction in Football”, Kaggle, February 2017.

[8]. Saif Uddin, “Football Match Prediction”, Kaggle, February 2020.

[9]. Gabriel II, “Expected Goals & Player Analysis”, Kaggle, November 2020.

[10]. Usama Waheed, “Expected Goals (xG) Model”, Kaggle, May 2023.