Machine Learning Report

"Football: Who Will Win?"

Léo ATTIG
leo.attig@edu.univ-eiffel.fr

Mathieu PERRIOT
mathieu.perriot@edu.univ-eiffel.fr

Sebastien SAUTIER
sebastien.sautier@edu.univ-eiffel.fr

Alexis TREMELLAT
alexis.tremellat@edu.univ-eiffel.fr

Master 2: SIA and SSIO
Data Science Competition on ChallengeData

January 2025

Project Supervisor

Dr. ALAOUI Jawad

Contents

1 Introduction & Problem understanding

Football is one of the most widely followed sports in the world, captivating millions of fans with its unpredictability and competitive nature. However, anticipating the outcome of football matches remains a challenging task due to the complexity of the game and the numerous variables involved. From player performance and team strategies to environmental factors such as weather and crowd influence, accurately forecasting match results requires sophisticated data analysis and machine learning techniques.

This project aims to leverage advanced data science methodologies to develop predictive models capable of determining the likely winners of football matches. By participating in the "Football: Who will win?" competition on ChallengeData, we will analyze match-related data, explore various modeling approaches, and fine-tune our models to achieve optimal performance. The primary goal is to build a robust predictive framework while gaining insight into the factors that influence the outcome of football matches.

In this report, we begin by exploring the data and addressing potential challenges, such as understanding the dataset. We then present a range of predictive models, compare their strengths and weaknesses, and discuss the strategies used for hyperparameter tuning and model validation. Finally, we analyze the results, reflect on the lessons learned, and suggest potential improvements for future work.

2 Data Exploration & Preprocessing

2.1 Dataset Description

The dataset provided for this project contains comprehensive statistics for football teams, divided into home and away performances. Key attributes include:

- Team-level metrics such as shots, passes, and saves.
- Match-level summaries including attacks, dangerous attacks, and goals.
- Contextual features like league and team names.

In total, the dataset spans 142 columns, encapsulating seasonal and recent-match summaries.

2.2 Data Loading and Initial Inspection

To begin the analysis, the data files were loaded using Python libraries such as Pandas and NumPy. The initial inspection revealed the following structure:

- Separate files for home and away team statistics.
- Target variable file specifying match outcomes.

The home and away statistics were merged into a unified dataset, ensuring alignment with the target variable for model training.

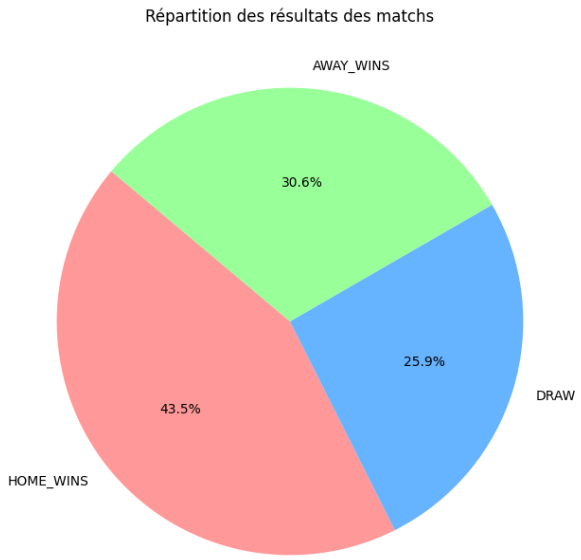

Figure 1: Repartition Results of Matchs

2.3 Data Cleaning and Transformation

Several preprocessing steps were performed:

- **Renaming Columns**: Columns were prefixed with `HOME_` or `AWAY_` to distinguish between home and away statistics.
- **Handling Missing Values**: Any occurrences of infinite values were replaced with NaN for subsequent imputation or removal.
- **Feature Concatenation**: Home and away statistics were concatenated, creating a comprehensive view for each match.

2.4 Exploratory Data Analysis

An initial exploration of the dataset provided insights into the distribution and variability of features. For instance:

- League-specific trends were observed, with variations in shooting, passing, and defensive metrics across teams.
- Recent-match statistics highlighted performance consistency and anomalies in team behavior.

Visualizations, such as histograms and correlation matrices, will be employed to further understand feature relationships and guide feature engineering for the predictive models.

3 Methodological Approaches with Pros & Cons

In this section, we present three different methodological approaches that exist in Machine Learning processes, detailing their advantages and drawbacks for each one of them.

3.1 Linear Models

Linear models, such as Logistic Regression, are foundational techniques in predictive modeling.

Advantages:

- **Simplicity and Interpretability**: Linear models are straightforward to implement and interpret, providing clear insights into feature importance.
- **Efficiency**: These models are computationally efficient, making them suitable for large datasets.
- **Well-suited for Linearly Separable Data**: They perform well when there is a linear relationship between input features and the target variable.

Disadvantages:

- **Limited Flexibility**: Linear models may struggle to capture complex, non-linear relationships in the data.
- **Sensitivity to Outliers**: They are prone to being influenced by outliers, which can skew results.

Reason to Consider or Discard:

We considered linear models as a baseline due to their simplicity and interpretability. However, their limited capacity to model non-linear patterns led us to explore more complex methods.

3.2 Tree-Based Methods

Tree-based methods, such as Decision Trees, Random Forests, and Gradient Boosting Machines (GBM), offer a more flexible approach.

Advantages:

- **Ability to Capture Non-Linear Relationships**: Tree-based methods can model complex interactions between features.
- **Robustness to Outliers**: They are less sensitive to outliers compared to linear models.
- **Feature Importance**: These methods provide insights into feature importance, helping to understand the contribution of each feature.

Disadvantages:

- **Risk of Overfitting**: Decision Trees, in particular, are prone to overfitting, though techniques like pruning or using ensemble methods (Random Forests, GBM) can mitigate this.
- **Higher Computational Cost**: Tree-based models, especially ensemble methods, require more computational resources and time for training.

Reason to Consider or Discard:

We included tree-based methods due to their flexibility and ability to handle non-linear relationships. Ensemble methods, like Random Forests and GBM, were particularly attractive due to their performance improvements over single decision trees.

3.3 Neural Networks

Neural networks, particularly deep learning models, have gained popularity for their powerful predictive capabilities.

Advantages:

- **High Predictive Power**: Neural networks can model highly complex and non-linear relationships.
- **Scalability**: They are capable of handling large-scale data and a vast number of features.
- **Automatic Feature Extraction**: Neural networks can automatically learn and extract important features from raw data.

Disadvantages:

- **High Computational Requirements**: Training neural networks is resource-intensive and requires significant computational power.
- **Lack of Interpretability**: The complex structure of neural networks makes them less interpretable compared to simpler models.
- **Need for Large Datasets**: Neural networks typically require large amounts of data to perform well, which might not always be available.

Reason to Consider or Discard:

Neural networks were considered due to their potential to achieve high accuracy. However, given the computational resources and time required, they were selected with caution and primarily used in scenarios where simpler models were insufficient.

4 Model Selection, Tuning & Validation

In this section, we detail the model selection, tuning, and validation process used to predict football match outcomes. We initially explored various classification approaches, each aiming to capture the dynamics of different match results. The following steps describe the approach taken and the reasons that led us to the final solution.

4.1 Initial Approach: Classic Classifiers

4.1.1 Random Tree Classifier

We first experimented with a **Random Tree Classifier**, a simple yet effective model for classification tasks. However, despite its simplicity, this model did not yield satisfactory results, mainly due to its limited ability to capture complex interactions between the features and its relatively low predictive power for this task. The results obtained were sub-par, and the performance was below expectation; this method provides us a 17% to 19% accuracy (which is worse than random guess.)

We attempted to perform feature selection by progressively choosing the top features. First, we selected the top 50 features, then the top 20, and finally the top 5. Interestingly, as we reduced the number of features, the prediction performance improved as follows:

- Top 50 features: 0.198
- Top 20 features: 0.205
- Top 5 features: 0.286

However, we decided to abandon this approach and proceeded to try XGBoost instead.

4.1.2 XGBoost

Next, we turned to **XGBoost**, a popular boosting algorithm that often performs well on complex tabular data. Although XGBoost showed slightly better results than the Random Tree Classifier, the performance was still not sufficient for building a reliable and robust model. Moreover, we observed that XGBoost tended to overfit when tuned too much on the training data. This overfitting led to poor performance on the validation set and reduced generalization ability.

We tested XGBoost with different feature selections as follows:

- Top 50 features: 0.2836
- Top 20 features: 0.2905
- Top 5 features: 0.3088

Despite these results, we decided to try a Convolutional Neural Network (CNN) approach to see if it could provide better performance.

4.1.3 CNN (Convolutional Neural Network)

As a more advanced approach, we also tested a **CNN (Convolutional Neural Network)** model for this task. While convolutional neural networks are typically used for image and spatiotemporal data, we explored this model for its potential to detect complex patterns in the data. However, despite showing some promise in learning rate plots and training curves, the model was unable to provide sufficiently strong performance. Like XGBoost, we observed significant overfitting, where the model specialized too much on the training data, sacrificing its ability to generalize to unseen data.

The model architecture consisted of several dense layers:

- The first dense layer with 256 units, activated by ReLU.
- A second dense layer with 128 units, also activated by ReLU.
- A third dense layer with 64 units, again activated by ReLU.
- The final output layer with a number of units equal to the number of possible classes, activated by softmax.

We compiled the model using the Adam optimizer, with categorical cross-entropy as the loss function and accuracy as the evaluation metric:

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(256, activation='relu', input_shape=(nb_inpiuts,)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(nb_outputs, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
training = model.fit(X_train, y_train, epochs=50, batch_size=32,
                     validation_data=(X_test, y_test))
```

The graph below shows the accuracy of the CNN model. It rapidly approached a performance of 1, indicating overfitting, as it specialized too much on the training set and failed to generalize well on the validation set.

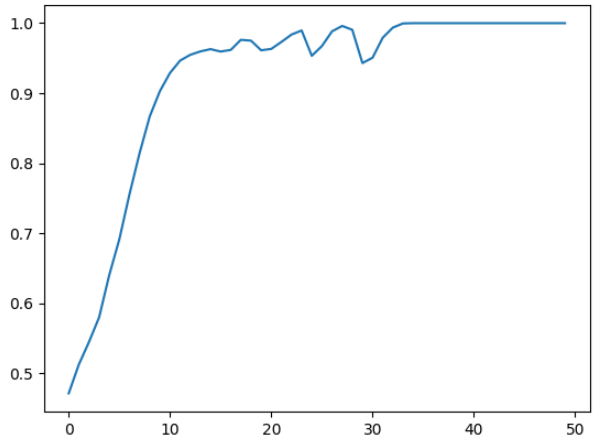

Figure 2: Convolutional Neural Networks accuracy

After training the model by selecting the top 50, 20, and 5 most important features, we obtained the following results:

- **Neural Network with Top 50 Features:** 0.396
- **Neural Network with Top 20 Features:** 0.401
- **Neural Network with Top 5 Features:** 0.438

The model achieved an accuracy of **43%** when predicting on new data. The accuracy showed slight improvements when using fewer features, but the overall performance was still not satisfactory for reliable predictions.

4.2 Feature Selection and Fine-Tuning

In an effort to improve the model's performance, we also explored the possibility of **feature selection**. We initially selected the most relevant features, using various techniques to identify those that contributed the most to the model's predictive power. Despite these efforts, we found that feature selection alone did not lead to substantial improvements in accuracy.

We also attempted to **fine-tune the models**, adjusting hyperparameters and training configurations to optimize performance. While this process led to minor improvements,

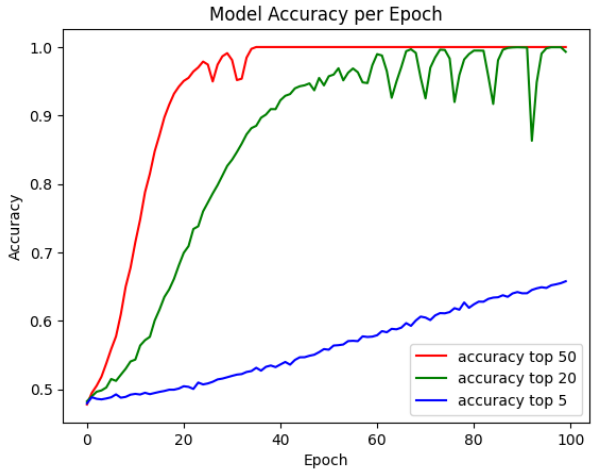

Figure 3: Convolutional Neural Networks accuracy with Features selection

none of the tuning efforts provided conclusive results. The models continued to exhibit either underfitting or overfitting, and no significant breakthroughs were achieved in this stage.

5 Final Chosen Solution & In-Depth Analysis

The final model chosen for predicting football match outcomes is based on three separate LightGBM models. These models were trained to predict three different possible outcomes for each match: **Home Wins**, **Draws**, and **Away Wins**. Each model provides the probability for each possible outcome, and we select the outcome with the highest probability, corresponding to the model most confident in its prediction. Below, we describe the key steps taken in developing and evaluating this approach.

5.1 Data Preparation

The dataset consists of statistics for both home and away teams, and was preprocessed as follows:

- **Loading Data**: Separate CSV files for home and away team statistics were loaded using `pandas`.
- **Feature Engineering**: Columns were prefixed with `HOME_` or `AWAY_` to clearly distinguish between the two sets of statistics for each team.
- **Data Concatenation**: The home and away team statistics were concatenated to create a unified dataset for each match.
- **Handling Missing Values**: Infinite values were replaced with NaN, allowing for imputation or removal as necessary before training.

5.2 Model Implementation

The model was implemented using the LightGBM library, with three separate models for predicting the three potential match outcomes: **Home Wins**, **Draws**, and **Away**

Wins. Each model is trained with the following configuration:

- **Parameter Settings**:
  - `boosting_type`: `gbdt`
  - `objective`: `multiclass`
  - `num_class`: 2
  - `metric`: `multi_logloss`
  - `num_leaves`: 31
  - `learning_rate`: 0.1
  - `feature_fraction`: 0.9
  - `bagging_fraction`: 0.8
  - `bagging_freq`: 5
  - `verbose`: 0
- **Training**: Each model is trained separately for predicting one of the three outcomes: Home Wins, Draws, and Away Wins. The dataset is split into training and validation sets, and the LightGBM model is trained using a large number of boosting rounds with early stopping after 100 rounds if no improvement is observed.
- **Evaluation**: The accuracy of each model is assessed using the `accuracy_score` metric. Each model provides the probability of the corresponding outcome, and we select the prediction of the model most confident in its decision. This is done by comparing the predicted probabilities and choosing the outcome with the highest probability for each match.

For each match, the three models are used to predict the probabilities of **Home Wins**, **Draws**, and **Away Wins**. The model with the highest probability for each match outcome is selected, and the corresponding predicted outcome (Home Win, Draw, or Away Win) is used as the final prediction. This method allows for robust and reliable predictions by leveraging the strengths of each model and selecting the most confident outcome.

In addition to the feature selection process, we also employed cross-validation methods to evaluate the model's generalization ability. Cross-validation is useful for assessing the model's performance by splitting the data into multiple subsets and training the model multiple times. This helps ensure that the model is not overfitting and provides more reliable performance estimates. We tested various cross-validation strategies, including `k-fold cross-validation`, to improve model robustness and prevent overfitting.

As shown in Figure 4, we explored different feature sets to improve the model's performance. Initially, we attempted selecting a subset of the top features for each model. For instance, we tested the performance of the model using the top 50 features, then the top 20, and finally the top 5 features. However, while reducing the number of features occasionally led to improvements, it was also observed that selecting fewer features often resulted in overfitting, and the overall performance did not improve consistently. In fact,

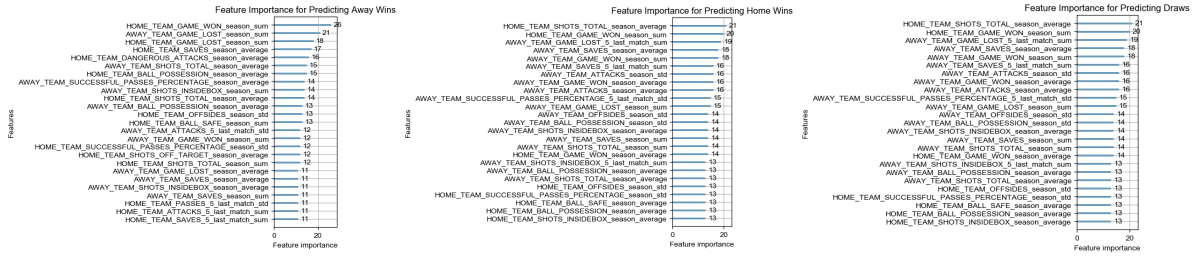

Figure 4: Feature Importance for the Three Models

no configuration with fewer features consistently outperformed the original model with all features.

This suggested that while some features were more important than others, removing too many features could harm the model's ability to generalize, which led us to the conclusion that a careful selection of features, combined with proper tuning of the models, would be necessary for achieving optimal results.

5.3 Prediction and Results

For each model, we obtained the following results:

- **Prediction for Away Wins**: 0.707
- **Prediction for Home Wins**: 0.6209
- **Prediction for Draws**: 0.7481

By taking the prediction with the highest confidence for each match, the final prediction accuracy on the challenge data was 0.4863, placing us in the 398th position.

5.4 Model Conclusion

In this study, we implemented a multi-model approach using LightGBM to predict the outcomes of football matches. Three models were trained to predict the probabilities for Home Wins, Draws, and Away Wins. The final prediction for each match was determined by selecting the outcome with the highest predicted probability.

Although our model achieved an accuracy of **0.4863** on the challenge data, ranking **398th**, this approach demonstrates the potential of combining multiple models to improve predictive performance. Further improvements can be made by fine-tuning the model parameters, incorporating additional features, or exploring other machine learning algorithms.

6 Conclusion & Lessons Learned

6.1 Conclusion

The aim of this project was to predict football match outcomes using machine learning techniques. After exploring various approaches, including linear models, tree-based methods, and neural networks, we opted for a multi-model approach with LightGBM. This strategy allowed us to capture the complex interactions between different game characteristics, yielding reliable predictions.

Although our model achieved an accuracy of 48.63% on the challenge data, it still fell short of the desired level of performance for fully reliable predictions. Despite this, the project demonstrates the potential of the chosen approach. To enhance the results in the future, improvements in feature selection, hyperparameter tuning, and potentially other machine learning algorithms would be necessary. This project highlighted the challenges of predicting football match outcomes while paving the way for future improvements.

6.2 Lessons Learned

One key lesson learned is that football match data is inherently complex, with many interdependent variables such as player statistics, team strategies, and external conditions. Proper understanding and cleaning of the data are essential before applying any machine learning models.

Additionally, feature selection proved to be a critical factor in improving model performance. While reducing the number of features in some cases led to better accuracy, overly restrictive selection resulted in a performance drop. This suggests that a more nuanced approach to selecting features might be beneficial.

The challenge of overfitting was also evident, particularly with models like neural networks and decision trees. These models showed signs of overfitting, reducing their ability to generalize effectively to new data. This highlights the importance of using techniques such as cross-validation and early stopping to mitigate this issue.

Furthermore, the use of a multi-category model (to predict home wins, draws, and away wins) was a useful approach for handling the complexity of match outcomes. However, this added an extra layer of complexity to the predictions.

Finally, looking ahead, there is room for further improvement. Adjustments to model parameters, exploring deeper neural networks, and incorporating additional data, such as player-specific information, could lead to better predictions in the future.