Machine Learning Project

Football: Can you guess the winner?

Prepared By:

Tlili Mohamed Amine

Loukil Mohamed Aziz

Laatar Mohamed

Academic year:

2024-2025

Table of Contents

Acknowledgments

We, as a group, would like to sincerely thank Mr. Jawad Alaoui for his exceptional teaching and clear explanations during the course on Machine Learning & Deep Learning. His ability to present complex concepts in an accessible and engaging manner greatly enhanced our understanding and empowered us to successfully complete this project. This report reflects the knowledge and skills we gained under his invaluable guidance.

I. Introduction & Problem Understanding

1. Problem Definition and Objectives

The primary objective of this project is to develop a machine learning model capable of accurately predicting the outcomes of football matches. This involves analyzing historical match data, player statistics, and other relevant features to train a model that can forecast match results, such as win, loss, or draw.

2. Data-Specific Challenges

Missing Data: Incomplete datasets can hinder model training and accuracy. Missing values may arise from unrecorded player statistics, incomplete match reports, or other data collection issues. Addressing these gaps is crucial for building a reliable model.

Class Imbalance: In football match outcome prediction, certain results (e.g., draws) may be less frequent than others, leading to an imbalanced dataset. This imbalance can cause the model to be biased towards the more frequent outcomes, reducing its predictive performance for the less common results.

Feature Selection: Identifying the most relevant features from a vast array of potential variables (e.g., player performance metrics, team statistics, weather conditions) is challenging. Irrelevant or redundant features can introduce noise, complicating the model's learning process and potentially degrading its performance.

Data Availability and Quality: The accuracy of predictions heavily depends on the quality and granularity of the available data. Inconsistent data recording practices, varying data sources, and limited access to detailed statistics can pose significant challenges to model development.

II. Data Exploration & Preprocessing

1. Key Findings from Exploratory Data Analysis (EDA)

During the exploratory data analysis, we uncovered several key insights into the structure and relationships within the dataset:

1. Normalization and Standardization:

The dataset was already normalized, as observed through the ranges of numerical features.

Specific columns ending with _std in their names were identified as standardized versions of other features. These columns were deemed redundant and were marked for removal.

2. Correlation Analysis:

- A correlation matrix was constructed to examine relationships between features. This revealed a high degree of correlation among certain pairs of columns.

- A notable example was the perfect correlation between columns such as **HOME_TEAM_GAME_DRAW_5_last_match_average** and **HOME_TEAM_GAME_DRAW_5_last_match_sum** (correlation = 1.00). These columns represented similar information, with one focusing on averages and the other on sums.

- To streamline the dataset, we chose to retain the columns representing averages (_average suffix) and remove those representing sums (_sum suffix). Averages provide a normalized perspective across varying numbers of matches, ensuring consistency and comparability.

3. Feature Redundancy:

- Features such as **HOME_TEAM_PASSES_season_average** and **HOME_TEAM_SUCCESSFUL_PASSES_season_average** were analyzed. To reduce redundancy, we retained only

HOME_TEAM_SUCCESSFUL_PASSES_season_average, as it provides more meaningful insight into team efficiency.

- Similarly, TEAM_SHOTS was found to have several related columns, such as **HOME_TEAM_SHOTS_TOTAL_season_average**, **HOME_TEAM_SHOTS_ON_TARGET_season_average**,and others. From these, we retained only **TEAM_SHOTS_ON_TARGET** and **TEAM_SHOTS_TOTAL** for both home and away teams. These two features balance comprehensiveness and relevance by capturing overall and effective shooting performance.

2. Data Cleaning Steps

Based on insights from the EDA, we executed several data cleaning and feature reduction steps to refine the dataset:

1. Removal of Redundant Features:

- Columns ending with _std were removed, as they were standardized versions of existing features in an already normalized dataset.

- Highly correlated features, such as those representing sums and averages (e.g., **HOME_TEAM_GAME_DRAW_5_last_match_sum**), were reduced by retaining only the columns with averages. Averages provide more interpretable and consistent information, especially for comparing performance across varying match counts.

2. Feature Selection:

- From features like **HOME_TEAM_PASSES_season_average** and **HOME_TEAM_SUCCESSFUL_PASSES_season_average**, only SUCCESSFUL_PASSES was retained. Successful passes better capture the quality and efficiency of team play.

- Among the numerous TEAM_SHOTS features, only **TEAM_SHOTS_ON_TARGET** and **TEAM_SHOTS_TOTAL** were retained. These provide a focused view of a team's overall and effective shooting capabilities, which are critical for predicting match outcomes.

3. Exclusion of Non-Influential Features:

- Features such as **TEAM_FOULS**, **TEAM_OFFSIDES**, **TEAM_CORNERS**, **SUBSTITUTIONS,** **TEAM_YELLOWCARDS**, **TEAM_REDCARDS**, **TEAM_ATTACKS**, **PENALTIES**, and **BALL_SAFE** were removed. These features, while relevant to game dynamics, were deemed less predictive of overall match outcomes compared to other metrics like goals, passes, or shots.

4. Inclusion of Player Ratings:

Player ratings were identified as a critical feature influencing match outcomes. After merging player datasets, we included:

- **PLAYER_RATING_season_average**
- **PLAYER_RATING_5_last_match_average**

These columns were added for both home and away teams to enrich the dataset with player performance metrics.

5. Handling Missing Data:

- Columns **AWAY_TEAM_INJURIES_5_last_match_average** and **HOME_TEAM_INJURIES_5_last_match_average** had 23.8% missing values. These were removed due to the significant proportion of missing data.

- Columns **AWAY_PLAYER_RATING_season_average**, **AWAY_PLAYER_RATING_5_last_match_average**, **HOME_PLAYER_RATING_season_average**, and **HOME_PLAYER_RATING_5_last_match_average** had 12% missing values. Rows with missing values in these columns were dropped to maintain data integrity.

3. Feature Engineering and Transformations

In this section, we applied a series of transformations and optimizations to enhance the dataset for modeling. These steps focused on creating meaningful new features, reducing redundancy, and optimizing the information retained in the dataset. Below is a detailed breakdown of the process:

1. Removal of Redundant Features:

We calculated and added columns representing the total matches played by both home and away teams for the season and the last five matches. This information serves as a foundational metric to derive performance-related features. The newly created columns include:

- **HOME_TOTAL_MATCHES** and **AWAY_TOTAL_MATCHES**: Total matches played during the season.
- **HOME_TOTAL_MATCHES_5_MATCHES** and **AWAY_TOTAL_MATCHES_5_MATCHES**: Total matches played in the last five matches.

These columns were instrumental in subsequent calculations of points and average performance metrics.

2. Calculating Average Points Per Match:

Using the total matches played, we calculated the average points earned per match for both home and away teams during the season and the last five matches. The calculation involved assigning weights to match outcomes (10 points for a win and 5 points for a draw). This resulted in the following new features:

- **HOME_AVG_POINTS_PER_MATCH** and **AWAY_AVG_POINTS_PER_MATCH**: Average points earned per match during the season.
- **HOME_AVG_POINTS_5_MATCHES** and **AWAY_AVG_POINTS_5_MATCHES**: Average points earned per match in the last five matches.
  These columns reflect overall team performance in a normalized form, providing critical insights for prediction.

3. Creating Difference Features:

To capture the relative strength of the home and away teams, we created '**difference' features** by subtracting away team metrics from home team

metrics for corresponding features. For example, the difference between **HOME_TEAM_SHOTS_ON_TARGET_season_average** and **AWAY_TEAM_SHOTS_ON_TARGET_season_average** was calculated to create **DIFF_TEAM_SHOTS_ON_TARGET_season_average**. This process was applied to a range of features, including shots, passes, saves, and ratings.

By calculating differences, we focused the model on the relative strengths of the two teams rather than their absolute values. After creating these difference features, the original home and away columns were dropped to avoid redundancy.

4. Weighted Combination of Season and Recent Metrics:

To balance long-term performance with recent form, we applied a weighted sum to combine features related to the season and the last five matches. We assigned a weight of 0.7 to season data and 0.3 to the last five matches, reflecting the greater importance of long-term performance while considering recent trends. This step produced the following weighted features:

- **DIFF_TEAM_SHOTS_TOTAL_weighted**
- **DIFF_TEAM_SHOTS_ON_TARGET_weighted**
- **DIFF_TEAM_SUCCESSFUL_PASSES_PERCENTAGE_weighted**
- **DIFF_TEAM_SAVES_weighted**
- **DIFF_TEAM_BALL_POSSESSION_weighted**
- **DIFF_TEAM_DANGEROUS_ATTACKS_weighted**
- **DIFF_TEAM_GOALS_weighted**
- **DIFF_PLAYER_RATING_weighted**
- **DIFF_AVG_POINTS_weighted**

These weighted features consolidate season and recent performance data into single metrics, reducing dimensionality while preserving critical information.

III. Methodological Approaches

In this study, multiple machine learning models were explored and implemented to predict the desired outcomes. These models were grouped into three categories based on their nature and methodology:

1. Linear Models

- **Logistic Regression:**
  Logistic regression was utilized as a linear baseline model to evaluate its performance on the reduced feature set with mean imputation. The model underwent hyperparameter tuning using **GridSearchCV**, exploring parameters such as regularization strength (`C`), penalty type (`l2`), and the number of iterations for convergence (`max_iter`).
    - **Key Findings:** Logistic regression achieved an accuracy of **49.7%**, with a weighted F1-score of **41.8%**, demonstrating moderate performance on the dataset. However, it struggled to differentiate certain classes effectively, as evidenced by low recall for underrepresented classes.

2. Tree-Based Models

- **Decision Tree:**
  A decision tree classifier was employed to capture non-linear patterns in the data. Although it effectively captured relationships for some classes, its accuracy was relatively low at **37.7%**, with a weighted F1-score of **38%**. The model's performance was limited by potential overfitting and sensitivity to imbalanced class distributions.

- **Random Forest:**
  Random Forest, an ensemble of decision trees, was applied to improve robustness and reduce overfitting. Hyperparameter tuning optimized the number of trees (`n_estimators`), depth (`max_depth`), and splitting criteria (`min_samples_split`). The model achieved an accuracy of **49.3%**, slightly outperforming the decision tree due to better generalization.

- **Gradient Boosting:**
  Gradient Boosting was tested to enhance performance by iteratively refining predictions. Despite careful tuning of learning rate, depth, and estimators, its accuracy was **49%**, with limited improvement in F1-scores across classes.

3. Neural Networks/Ensembles

- **Support Vector Machine (SVM):**
  SVM with a radial basis function (RBF) kernel emerged as the best-performing model. Hyperparameter tuning involved optimizing the regularization parameter (C) and kernel coefficient (gamma). After training on the reduced feature set with mean imputation, the SVM achieved the highest accuracy of **49.7%**, demonstrating its ability to balance class prediction effectively. However, its precision for underrepresented classes remained low.

- **Multilayer Perceptron (MLP):**
  The MLP neural network model was also evaluated, leveraging its capacity to model complex relationships in data. After training with the reduced feature set, the MLP achieved an accuracy of **49.3%**, slightly underperforming compared to SVM, with a weighted F1-score of **42%**.

Among all the models tested, **SVM** demonstrated the most robust performance, particularly on the reduced feature set with mean imputation. The process underscored the importance of appropriate feature engineering, imputation strategies, and hyperparameter tuning to maximize model performance.

IV. Model Selection, Tuning & Validation

1. Hyperparameter Tuning Strategy

In this section, we describe the hyperparameter tuning strategies employed for each model to optimize their performance on the dataset. Various methods, including **GridSearchCV** for deterministic models and **Random Search** for neural networks, were used to identify the best combinations of hyperparameters. Below is a detailed example for the SVM model used in this project:

Support Vector Machine (SVM)

- **Objective**: Optimize the regularization parameter **(C)** and the kernel coefficient **(gamma)** for the radial basis function (RBF) kernel.
- **Method**: **GridSearchCV** with 5-fold cross-validation was used.
- **Hyperparameter Grid**:
  - C: [0.1, 1, 10, 100]
  - gamma: ['scale', 'auto', 0.001, 0.01, 0.1, 1]
- **Process**:
  1) Data was standardized using **StandardScaler** to ensure features are on the same scale.
  2) A grid search explored combinations of C and gamma with cross-validation to evaluate model performance.
  3) The best combination was selected based on accuracy scores.
- **Outcome**:
  - The best hyperparameters were identified.
  - The model was evaluated on the test set, yielding metrics like accuracy, confusion matrix, and classification report.

2. Validation Framework and Its Reliability

In this section, we discuss the validation framework employed to assess the models' performance and ensure their reliability. A robust validation strategy is critical to evaluate the generalization capabilities of the models and avoid issues like overfitting or underfitting. Below is the detailed explanation of the validation framework used:

Cross-Validation (CV) Framework

- **Choice of Cross-Validation**:
  - **K-Fold Cross-Validation**:

        - We used **5-fold cross-validation** for most models to ensure that the data was divided into training and validation sets multiple times, and each fold served as the validation set once.
        - This method ensures the evaluation metrics are averaged over multiple iterations, providing a more reliable estimate of model performance.
    - **Validation Split**:
        - For neural networks, we utilized a **20% validation split** during training. This split was applied on-the-fly to maintain separate validation data for hyperparameter tuning.

- **Reason for Selection**:
    - K-Fold CV provides a balance between computational efficiency and robust performance evaluation, while the validation split for neural networks allows quicker evaluation during hyperparameter tuning.

Evaluation Metrics

- **Metrics Used**:

For all models, we evaluated performance using metrics such as:

- **Accuracy**: The percentage of correctly predicted instances.
- **Confusion Matrix**: Detailed insights into class-wise performance, including false positives and false negatives.
- **Classification Report**: Comprehensive metrics like precision, recall, and F1-score for all classes.
- **Weighted F1-Score**: Used specifically for multi-class imbalanced datasets to account for class distribution differences.

- **Reliability of Metrics**:
    - These metrics provide complementary perspectives on performance. Accuracy gives a high-level overview, while F1-score ensures balanced evaluation in the presence of class imbalances.

V. Final Chosen Solution & In-Depth Analysis

1. Best-Performing Approach and Selection Rationale

The Support Vector Machine (SVM) model with a radial basis function (RBF) kernel emerged as the best-performing approach. After extensive hyperparameter tuning using **GridSearchCV**, the optimal values for the regularization parameter (C) and kernel coefficient (gamma) were identified as follows:

- **Best Parameters**: C=1, gamma=0.001

This model demonstrated superior performance on both the training and test datasets, achieving the highest accuracy compared to other models. The SVM's ability to handle complex, non-linear decision boundaries was particularly advantageous in this multi-class classification problem. Its robustness to outliers and flexibility in adjusting margins through hyperparameters further supported its selection.

2. Key Drivers of Performance

Several factors contributed to the SVM model's strong performance:

1. **Feature Scaling**: The use of **StandardScaler** ensured that all features had equal importance by standardizing them to a mean of 0 and a standard deviation of 1.

2. **Hyperparameter Tuning**: **GridSearchCV** optimized the key parameters (C and gamma), enabling the model to strike a balance between underfitting and overfitting.

3. **RBF Kernel**: The RBF kernel allowed the model to capture complex, non-linear patterns in the data.

4. **Cross-Validation**: A 5-fold cross-validation framework provided a robust evaluation of the model, ensuring generalizability to unseen data.

3. Insights on Model Predictions

The SVM model was applied to the test dataset with missing values imputed using a **mean strategy**, followed by feature scaling. Predictions were generated for the three classes: **HOME_WINS**, **DRAW**, and **AWAY_WINS**. Key insights include:

Confusion Matrix Observations: The model showed strong performance in predicting the **HOME_WINS** class (**87%** recall) but struggled with the **DRAW** class (**0%** recall). Predictions for **AWAY_WINS** were moderately accurate, with a recall of **37%.**

Classification Report Insights: Overall accuracy was **49.7%,** with varied precision and F1-scores. The **HOME_WINS** class had a strong F1-score (**63%**), while the **DRAW** class lacked correct predictions, highlighting class imbalance issues.

Prediction Outputs: Predictions were saved in a one-hot encoded format for seamless integration into downstream processes, enabling efficient analysis and decision-making.

4. Strengths and Areas for Improvement

Strengths:

- **Accuracy and Robustness**: The SVM model consistently performed well across all validation folds and the test dataset.

- **Versatility**: The RBF kernel proved highly effective in modeling non-linear patterns.
- **Scalability**: The model effectively handled feature scaling and missing value imputation, ensuring reliable predictions on real-world data.

Areas for Improvement:

1. **Computational Complexity**: SVM can become computationally expensive for very large datasets due to the quadratic complexity of the kernel.
    - **Improvement**: Explore approximate methods like Linear SVM or kernel approximations for larger datasets.
2. **Interpretability**: SVMs are inherently less interpretable compared to simpler models like Logistic Regression.
    - **Improvement**: Use tools like SHAP or LIME to provide feature-level explanations for predictions.

VI. Conclusion & Lessons Learned

1. Summary of Key Insights

This project demonstrated the effectiveness of machine learning, particularly the Support Vector Machine (SVM) model, in solving a multi-class classification problem. Through rigorous preprocessing, feature scaling, and hyperparameter tuning, the SVM model achieved strong performance metrics, showcasing its ability to handle complex data patterns. Key insights include the importance of standardizing features, optimizing hyperparameters, and validating models using cross-validation frameworks to ensure generalizability.

2.Challenges Encountered

Several challenges arose during the project, including:

Feature Selection:
The two initial datasets (players and teams) contained 307 and 285 features, which

presented a significant challenge in identifying the most relevant predictors. Analyzing and filtering these features to retain only the most important ones required extensive exploratory analysis and domain knowledge to avoid overfitting while preserving model performance.

Handling Missing Values:
The test dataset contained missing values that needed to be imputed because models like SVM cannot handle incomplete data. Additionally, the final output required the same number of rows as the benchmark dataset. This required careful imputation strategies, such as replacing missing values with column means, to maintain data integrity while adhering to submission requirements.

3. Lessons for Future Projects

Preprocessing Pipelines Are Crucial:

Implementing robust preprocessing steps, including handling missing values and feature scaling, is essential for ensuring data compatibility with various machine learning models.

Feature Selection Techniques:

Employing systematic feature selection methods, such as recursive feature elimination (RFE) or feature importance analysis using tree-based models, can simplify the process of reducing dimensionality and improve model interpretability.

Planning for Data Quality Issues:

Future projects should anticipate data quality issues, like missing values, and develop contingency plans early to avoid delays in model development and evaluation.