1. Introduction

Predicting the outcomes of European football matches, deciding whether a team wins, draws, or loses, by using data science and machine learning, is a complex task.

For my 'final work' at Erasmushogeschool Brussel, studying applied informatics, I took it upon myself to take a deep dive into this daunting issue, aiming on achieving comparable results than existing models. My initial goal was to produce a model that could achieve a steady average accuracy of at least 75%. Besides the model's performance, as an aspiring data scientist, the main priority of my final work was the AI knowledge acquisition itself.

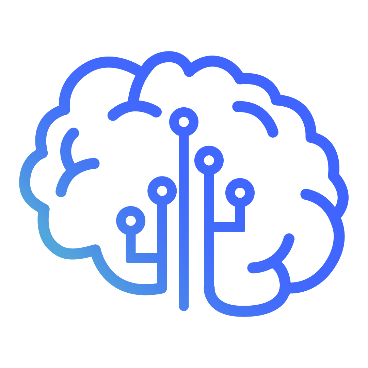
The process of developing a comprehensive model, built in Jupyter Notebook using Python, and performing at a satisfying level towards my preset goals, turned out to be an iterative process of analysing, developing and testing, and proved to be a challenging endeavour at times frustrating and intimidating.

Starting from a baseline model with minimal hyperparameter tuning, and without any feature engineering whatsoever, the progress made by extensive data analysis and thorough feature development was impressive.

The final outcome of my work is an ensemble[[1]](#footnote-1) model, which I named 'football brain', contrived of 7 distinctive classification models, performing in a binary classification strategy, displaying an average accuracy of around 83%, surpassing the expectations of myself and the people I shared my findings with.

An important note to make is that the development of this project will not cease after finishing my degree. The project will continue to be improved in the future. New engineering techniques will be tested, datasets will continue to be analysed, improved and expanded, and the path of moving the whole project to an 'MLOps[[2]](#footnote-2)' environment will be investigated.

Happy to share that during the process of developing the project, interest has grown from an independent investor, who is interested in monetizing the project by packaging it into a marketable online application.



1. Methodology
   1. Data Collection

During the initial phase of development, the collection of data was a critical task, undertaken by scraping from the football statistics website www.fbref.com. FBref, renowned for its comprehensive tracking of football team and player statistics globally, presented a significant source of valuable data. However, the website did not offer a straightforward method for downloading data in CSV format. Consequently, an intricate web scraper was constructed in Jupyter Notebook, designed to gather match data on a weekly basis. This process involved considerable effort to ensure the accuracy and reliability of the scraped data.

Upon thorough evaluation and deliberation, it became evident that the datasets obtained from FBref.com were insufficient for my needs. The limitations of the data, coupled with the complexities involved in maintaining the scraper, led to the decision to discontinue using these datasets.

A screenshot of a web page

Description automatically generatedIn the search for a more suitable data source, I discovered www.football-data.co.uk, a website that not only mirrored the statistical depth of FBref.com but also offered additional insights. Notably, next to offering ready to download CSV files for competitions of over 25 countries, this alternative source provided bookmakers' odds for both past and upcoming matches. These odds are instrumental in identifying which predictions have the potential to be most profitable, thereby enhancing the predictive models' effectiveness. The availability of such critical data points could significantly influence the accuracy and financial viability of our predictions.

* 1. Types of Data Collected

The datasets provided by football-data.co.uk encompass a vast amount of information collected for every match. However, not all of this data is relevant to the project. Therefore, only a selection of the data that is most pertinent to our analysis is used.

Div = League Division

Date = Match Date (dd/mm/yy)

Time = Time of match kick off

HomeTeam = Home Team

AwayTeam = Away Team

FTHG = Full Time Home Team Goals

FTAG = Full Time Away Team Goals

FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win)

HTHG = Half Time Home Team Goals

HTAG = Half Time Away Team Goals

HTR = Half Time Result (H=Home Win, D=Draw, A=Away Win)

HS = Home Team Shots

AS = Away Team Shots

HST = Home Team Shots on Target

AST = Away Team Shots on Target

* 1. Data Cleaning and Preprocessing

The ensemble model created in this project is centered around various Gradient Boosting models because of their speed and performance. Internally, gradient boosting models represent all problems as a regression predictive modeling problem that only takes numerical values as input. If the data is in a different form, it must be prepared into the expected format.

Dates are converted to integers through a function, and all categorical data (team names, competion names) are converted to dictionary values so they can be translated back to their original form after training.  
  


* 1. Handling Missing Data

The limited subset of data used from the datasets provided by football-data.co.uk has little to no missing information. The only columns with a minimal amount of missing data were HS, AS, HST, and AST. To address these gaps, I employed a technique called imputation, where the empty values in these columns are filled with their respective means. This method ensured that my dataset remained complete and reliable for analysis and training.

* 1. Data Engineering and Feature selection

Data engineering is undoubtedly the aspect of my project that demanded the most significant investment of development time. This critical phase involved intricate processes of collecting, cleaning, transforming, and organizing data to ensure its quality and usability for analysis. Given the complexities and challenges associated with managing large datasets, this stage required meticulous attention to detail, robust problem-solving skills, and a deep understanding of data architectures and tools.

From the outset, it was pivotal that the model performed equally well on test data as well as on unseen validation data. Ensuring consistent performance across different datasets was essential for the model’s reliability and accuracy. To achieve this, I deemed it crucial to engineer new features based solely on data available before a match starts. This approach prevented any potential data leakage and ensured that the model's predictions were based on realistic and actionable information. By adhering to this principle, I maintained the integrity of the model's performance and provided a robust foundation for its application in real-world scenarios.

* + 1. Team Form

To evaluate each team's current form, we calculate the average points earned over the past five games. This metric provides a clear, dynamic view of recent performance, highlighting trends in the team's effectiveness. It offers insights into their current momentum and potential future outcomes.

* + 1. ELO rating

An ELO rating applied to football is a method of ranking teams based on their performance in matches, similar to its original use in chess. This system assigns each team a numerical rating that reflects its overall strength. The ELO rating starts with a base score for all teams and adjusts after each game based on the match's outcome, the ratings of the opponents, and the expected outcome. When a team wins, it gains points, and the losing team loses points. The magnitude of the point change depends on the expected result; beating a higher-rated team yields more points than beating a lower-rated one. Conversely, losing to a lower-rated team results in a more significant point loss. The ELO system is dynamic, updating continuously as teams play more games, which allows it to adapt to changes in team strength over time. By incorporating factors like home-field advantage and goal differentials, the ELO rating provides a nuanced and quantitative measure of team performance, making it a valuable feature for predicting future match outcomes in football.

A table with numbers and names

Description automatically generated

1Calculated ELO rating of teams in the European Top 5 competitions, as of May 20th 2024

* + 1. Average points per game

To evaluate each team's overall performance in the current season, the average points earned per game is calculated weekly. This metric offers a clear, dynamic view of how each team progresses and performs throughout the season.

* + 1. Average goals scored/conceded per game

In addition to calculating the average points earned per game in the current season, we also determine the average goals scored and conceded per game. This comprehensive analysis provides a deeper insight into a team's offensive and defensive performance, allowing for a more nuanced understanding of their overall effectiveness on the field. By evaluating both points and goals, we can better assess the strengths and weaknesses of each team throughout the season.History versus an opponent

The weighted points a team earns against another team, calculated over their past five matches, are determined with the most recent match carrying the greatest weight.

* + 1. Rolling averages

For several columns in the dataset, including HS, AS, HST, and AST[[3]](#footnote-3), rolling averages are calculated over the past five matches. This involves taking the average of a specified number of data points, continuously updated by adding the most recent data point and removing the oldest. By doing so, rolling averages smooth out short-term fluctuations and highlight longer-term trends, providing a clearer view of performance over time.

* + 1. Expected Goals (XG)

Expected Goals (xG) is a metric used to evaluate the quality of scoring chances for both teams in a match. It estimates the likelihood of a shot resulting in a goal based on factors such as a team’s shots on target and the goals scored in a match. The xG metric provides a more nuanced understanding of a team's offensive effectiveness.

* + 1. Team vs Opponent metric gaps

During final testing, I discovered that creating additional features to highlight the differences between a team's metrics and those of their opponent positively impacted the model's accuracy. These gaps were calculated for key metrics, including ELO, Points, xG, and Form.

The final result is a dataset composed of the aforementioned engineered metrics, supplemented with the average bookmaker's odds for Win/Draw/Lose. During feature selection, only features that positively impacted the model's accuracy were retained, while those with little to no added value were removed. This enriched dataset encapsulates various performance indicators, offering a comprehensive view of each team's strengths, weaknesses, and trends. By integrating these detailed metrics, the dataset serves as a robust foundation for further analysis and predictive modeling.

1. Model Development

After thorough analysis and experimenting with a variety of model combinations, I ultimately chose to work with seven distinct classification models, as they delivered the most promising results. Interestingly, some models that were anticipated to excel did not meet expectations, while others, unexpectedly, outperformed them.

* 1. Model 1: XGBoost Classifier

In the thrilling world of football match prediction, the XGBoostClassifier has emerged as a powerful tool within ensemble models. XGBoost, short for eXtreme Gradient Boosting, is celebrated for its ability to handle large datasets and deliver high predictive accuracy. Developed by Tianqi Chen, it builds upon the principles of gradient boosting, an ensemble technique that constructs multiple weak learners, usually decision trees, and combines them to form a strong predictive model.

A diagram of a tree

Description automatically generatedOne of XGBoost’s most compelling strengths is its efficiency and speed. It utilizes advanced regularization techniques to prevent overfitting, ensuring robust performance even with complex data. Furthermore, its scalability allows it to tackle extensive datasets common in football match predictions. This is crucial when considering the myriad of variables such as player statistics, weather conditions, and team strategies.

However, XGBoost is not without its limitations. The model's complexity can be a double-edged sword; it requires careful parameter tuning and can be computationally intensive. Additionally, it demands a significant amount of memory, which can be a drawback for some applications.

In the context of predicting football match outcomes, XGBoostClassifier shines when integrated into an ensemble model. It leverages its strengths to synthesize diverse data points, offering predictions that are both accurate and insightful. Nonetheless, practitioners must be mindful of its resource requirements and the need for meticulous optimization to fully harness its potential. In summary, while XGBoostClassifier is a formidable asset in football analytics, its application requires a blend of computational power and expert tuning.

* 1. Model 2: Gradient Boosting Classifier

In the realm of sports analytics, particularly when predicting football match outcomes, the GradientBoostingClassifier (GBC) stands out as a powerful tool. This model, part of an ensemble approach, excels in its ability to enhance predictive accuracy by iteratively correcting errors from weaker models. Its strength lies in its robustness and adaptability, making it particularly effective for complex datasets with diverse features.

However, the GradientBoostingClassifier is not without its drawbacks. Training can be computationally intensive, demanding significant time and resources, which can be a limiting factor for real-time applications. Moreover, it is prone to overfitting, especially with small A diagram of a model boosting

Description automatically generateddatasets, necessitating careful tuning and validation.

In our football match prediction model, GBC plays a crucial role. By leveraging past match data, player statistics, and other relevant factors, it helps create a comprehensive and nuanced predictive framework. This ensemble method, combining multiple GBC models, ensures a higher degree of accuracy and reliability, aiding analysts and enthusiasts alike in making informed predictions.

While the GradientBoostingClassifier is undeniably potent, balancing its strengths and weaknesses through meticulous implementation is key to unlocking its full potential in sports analytics.

* 1. Model 3: Logistic Regression

In the realm of predicting football match outcomes, the Logistic Regression model stands as a robust cornerstone within ensemble methods. Originally developed as a binary classifier, Logistic Regression estimates the probability of a categorical outcome, making it particularly adept at scenarios where the result is win or lose, or in some cases, win, lose, or draw.

One of the primary strengths of Logistic Regression lies in its simplicity and interpretability. It offers clear insights into the impact of each feature on the prediction, which is invaluable for understanding the dynamics of football matches. Furthermore, its computational efficiency ensures quick training times, a vital attribute when dealing with large datasets or when integrating real-time data.

A diagram of a error

Description automatically generatedHowever, this model is not without its limitations. Logistic Regression assumes a linear relationship between the independent variables and the log odds of the dependent variable. In the complex world of football, where interactions between variables can be highly non-linear, this assumption can sometimes lead to suboptimal performance. Additionally, Logistic Regression can struggle with multicollinearity and may require careful preprocessing of data to mitigate these effects.

Despite these weaknesses, Logistic Regression's role in ensemble models for predicting football outcomes is significant. When combined with other models, it helps to capture a diverse range of patterns in the data, enhancing overall predictive performance. By leveraging its strengths and compensating for its weaknesses through ensemble techniques, Logistic Regression contributes to more accurate and reliable match predictions, guiding fans and analysts alike in their quest for foresight into the beautiful game.

* 1. Model 4: Random Forest Classifier

In the fascinating realm of football match prediction, the RandomForestClassifier shines as a pivotal component of our ensemble model. Originating from the decision tree family, this model excels by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks. The inherent robustness of RandomForestClassifier lies in its ability to handle a vast number of features and its resistance to overfitting. This resilience is achieved through bootstrapping and aggregation, which diversify the trees and enhance generalization.

The strengths of RandomForestClassifier are manifold. It is highly effective with large datasets and complex feature interactions, making it a perfect fit for football match prediction, where numerous variables interplay. Moreover, it provides insights into feature importance, helping us understand which factors significantly influence match outcomes.

A diagram of a tree

Description automatically generatedHowever, the model is not without its weaknesses. It can be computationally intensive and slower to predict compared to simpler models, especially with large forests. Additionally, it might not perform well with data that has little variation or too many irrelevant features, necessitating careful preprocessing.

Despite these limitations, the RandomForestClassifier's performance in our ensemble approach significantly boosts predictive accuracy. By combining it with other models, we harness its strengths while mitigating its weaknesses, leading to a well-rounded, robust predictive system. This amalgamation not only improves match outcome predictions but also provides a deeper understanding of the factors driving these results, ultimately offering a powerful tool for football analytics.

* 1. Model 5: Adaboost Classifier

The AdaBoostClassifier, an ensemble learning technique, has proven to be a powerful tool in predicting football match outcomes. This model excels by combining the strengths of multiple weak classifiers to form a robust predictive model. By sequentially focusing on misclassified instances, AdaBoostClassifier effectively reduces bias and variance, leading to more accurate predictions. This characteristic is particularly beneficial in the unpredictable domain of sports, where numerous variables influence the results.

One of the key strengths of AdaBoostClassifier lies in its ability to enhance performance through boosting. By iteratively adjusting the weights of misclassified samples, it hones in on the most challenging instances, thereby improving the overall accuracy. Additionally, its flexibility to work with various weak learners makes it adaptable and versatile.

A diagram of a tree

Description automatically generatedHowever, AdaBoostClassifier is not without its weaknesses. It is sensitive to noisy data and outliers, which can lead to overfitting. This sensitivity requires careful data preprocessing and sometimes limits its effectiveness in highly variable datasets typical of football matches. Moreover, the performance of AdaBoost can be computationally intensive, especially with large datasets, potentially leading to longer training times.

In the context of football match prediction, the AdaBoostClassifier demonstrates notable promise. Its capacity to refine predictions by addressing previous errors makes it a valuable component of ensemble models designed for this purpose. When combined with other models, it can significantly enhance the accuracy of match outcome predictions, providing valuable insights for analysts and enthusiasts alike.

In summary, while AdaBoostClassifier offers robust performance and adaptability, it must be handled with care due to its sensitivity to noisy data and computational demands. Its strategic use in ensemble models can unlock profound predictive capabilities in the dynamic world of football match forecasting.

* 1. Model 6: LGBM Classifier

A diagram of a diagram

Description automatically generatedWhen it comes to predicting the outcomes of football matches, the LGBMClassifier (LightGBM Classifier) stands out as a robust choice within ensemble models. Originating from Microsoft’s LightGBM framework, this model leverages gradient boosting techniques to offer superior performance, particularly with large datasets.

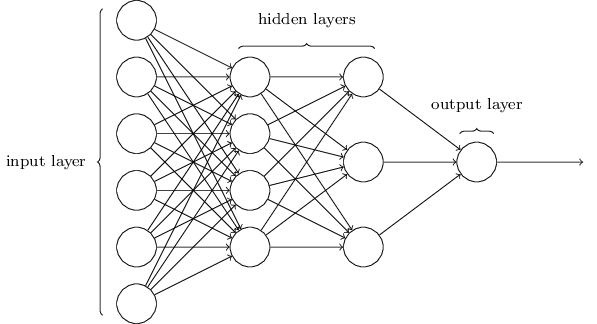
One of the primary strengths of the LGBMClassifier is its speed and efficiency. It outpaces many traditional algorithms thanks to its ability to handle massive data volumes with low memory usage. Additionally, it boasts remarkable accuracy and scalability, making it a favorite for complex predictive tasks like sports outcomes.

However, the LGBMClassifier isn't without its drawbacks. It can be sensitive to overfitting, especially if hyperparameters are not meticulously tuned. Moreover, the model's interpretability is limited compared to simpler algorithms, posing challenges in understanding the rationale behind its predictions.

In ensemble settings, combining the LGBMClassifier with other models can mitigate some of its weaknesses, enhancing overall prediction accuracy. By blending the strengths of multiple algorithms, the ensemble approach provides a more balanced and reliable prediction framework, crucial for the dynamic and unpredictable nature of football matches. Thus, the LGBMClassifier, with its blend of speed and precision, plays a pivotal role in pushing the boundaries of predictive modeling in sports analytics.

* 1. Model 7: MLP Classifier

When it comes to predicting football match outcomes, one standout component of our ensemble model is the Multi-Layer Perceptron Classifier (MLPClassifier). This neural network model is particularly adept at capturing non-linear patterns in data due to its architecture, which consists of multiple layers of nodes. Each node, or neuron, in these layers uses a non-linear activation function to learn complex relationships within the input data.

The MLPClassifier excels in scenarios where the relationships between features are intricate and not immediately apparent. Its ability to handle high-dimensional data and learn from a wide array of features makes it invaluable in the dynamic and often unpredictable realm of football match predictions. However, this model is not without its drawbacks. Training an MLPClassifier can be computationally intensive and time-consuming, requiring significant processing power and large datasets to achieve optimal performance. Additionally, it is prone to overfitting, especially if the model is too complex or the training data is not sufficiently large or diverse.

In our ensemble model, the MLPClassifier's strengths are harnessed alongside other algorithms to improve overall predictive accuracy. By combining the MLPClassifier with models that excel in other areas, such as decision trees or logistic regression, we mitigate its weaknesses and capitalize on its unique capabilities. This synergistic approach ensures a more robust and reliable prediction framework, ultimately leading to more accurate football match outcomes.

Thus, the MLPClassifier, with its strengths and occasional limitations, plays a pivotal role in our predictive ensemble, illustrating the power and necessity of diverse algorithmic approaches in machine learning.

* 1. Hyperparameter Tuning

In developing the ensemble model, I focused on hyperparameter tuning for four of the seven models: XGBoost, Random Forest, Logistic Regression, and Gradient Boosting. Through extensive testing across various scenarios, this particular configuration emerged as the most effective, consistently yielding the best results.

Hyperparameter tuning is a critical step that can significantly impact the performance of machine learning models. By fine-tuning these parameters, we can enhance the model's accuracy and generalization capabilities. For XGBoost and Gradient Boosting, adjusting parameters like learning rate and max depth helped in capturing complex patterns without overfitting. In the case of Random Forest, parameters such as the number of estimators and max features played a crucial role in balancing bias and variance. Logistic Regression, though simpler, benefited from tuning the regularization strength to prevent overfitting while maintaining interpretability.

Each model contributes uniquely to the ensemble, and optimizing their hyperparameters ensures that we leverage their strengths to the fullest. This approach allows the ensemble to make more robust and reliable predictions. Ultimately, the time and effort invested in this meticulous tuning process paid off, as evidenced by the superior performance of the ensemble model in predicting football match outcomes.



* + 1. XGBoost Classifier
* **max\_depth:** This sets the maximum depth of each tree. We use values from 1 to 6. Greater depth can capture complex patterns but may lead to overfitting.
* **learning\_rate:** Also known as eta, this parameter controls how much each tree contributes to the model. Values of 0.05, 0.1, and 0.15 help balance accuracy and training speed. Lower rates slow learning for finer adjustments.
* **reg\_lambda:** This is the L2 regularization term, adding a penalty proportional to the square of the weights. We test 0.01 and 0.1. Higher lambda values simplify the model by constraining weights.
* **alpha:** The L1 regularization term adds a penalty equal to the absolute value of weights, promoting sparsity. Values of 0, 0.5, and 1 are tested, with higher values enforcing more sparsity.
* **colsample\_bytree:** This parameter specifies the fraction of features sampled for each tree. We use 0.7 and 0.9 to enhance model robustness by introducing feature diversity.
* **subsample:** This parameter controls the fraction of training data used to fit each tree. Values of 0.75 and 0.85 help reduce overfitting by making the model more robust to data variations.
* **n\_estimators:** This indicates the number of trees in the model. We consider 1, 5, and 10 trees. More trees can improve accuracy but also risk overfitting and increase computational cost.
  + 1. Random Forest Classifier
* **max\_depth:** This parameter defines the maximum depth of each tree in the forest. We explore values of 1 and 2. A deeper tree can capture more detailed patterns but may overfit the training data. A shallower tree, on the other hand, ensures the model remains simple and generalizes better.
* **min\_samples\_split:** This is the minimum number of samples required to split an internal node. By testing values of 3 and 6, we control the growth of the tree. Higher values prevent the model from learning overly specific patterns, thereby reducing overfitting.
* **min\_samples\_leaf:** This parameter determines the minimum number of samples that a leaf node must have. We consider values of 1 and 3. Setting this parameter helps ensure that leaves have sufficient data, which improves the stability and generalization of the model.
* **n\_estimators:** This denotes the number of trees in the forest. We test 1, 5, and 10 trees. More trees generally lead to better performance, as the model can average out more predictions, but this comes at the cost of increased computational complexity and risk of overfitting.
* **max\_features:** This parameter indicates the number of features to consider when looking for the best split. We use ‘sqrt’ and ‘log2’. ‘Sqrt’ takes the square root of the total number of features, while ‘log2’ takes the logarithm base 2 of the total features. These methods reduce variance and improve the model's robustness.
  + 1. Logistic Regression
* **C:** This controls regularization strength. Smaller values (0.1) enforce stronger regularization, simplifying the model and reducing overfitting, while larger values (1) make the model more complex by reducing regularization.
* **penalty:** Specifies the norm for penalization. 'l1' promotes sparsity by using absolute values, 'l2' reduces complexity with squared values, and 'elasticnet' combines both, handling correlated features effectively.
* **solver:** The optimization algorithm used. 'saga' supports both L1 and L2 regularization, and is efficient for large, sparse datasets, making it versatile for our needs.
* **l1\_ratio:** Used with 'elasticnet' penalty, it balances L1 and L2 regularization. We set it to 0.5 for an equal mix, allowing fine-tuning between sparsity and smoothness.
* **class\_weight:** Adjusts weights for imbalanced datasets. 'balanced' automatically weights classes inversely proportional to their frequencies, ensuring equal attention to all classes.
  + 1. Gradient Boosting
* **learning\_rate:** Determines how much each tree contributes to the final model. We test 0.01, 0.1, and 0.15. Lower rates slow learning for finer adjustments, helping to avoid overfitting.
* **n\_estimators:** Indicates the number of boosting stages (trees). We use 1, 5, and 10. More trees can improve accuracy but may increase overfitting and computational cost.
* **max\_depth:** Sets the maximum depth of each tree. We test depths of 3, 5, and 7. Deeper trees capture complex patterns but can overfit if too deep.
* **min\_samples\_split:** Controls the minimum samples required to split a node. We use values of 2 and 5. Higher values prevent splits on noisy data, improving robustness.
* **min\_samples\_leaf:** Defines the minimum samples required at a leaf node. We test 1 and 2. This helps prevent overfitting by ensuring leaves have enough samples.
  + 1. Randomized Search Cross Validation

A diagram of a randomized search

Description automatically generatedRandomized search is a hyperparameter optimization technique that randomly samples from a specified distribution of parameters. It works by selecting random combinations of hyperparameters from the given distributions, evaluating the performance of each combination, and identifying the best-performing set. I used RandomizedSearchCV to efficiently explore a wide range of hyperparameter combinations for the ensemble model without the exhaustive computation required by grid search. This method randomly samples different parameter values, evaluating each combination to identify the best-performing set. It balances thorough exploration of the hyperparameter space with computational efficiency, making it ideal for finding optimal model settings within a reasonable time frame.



* **estimator:** `ensemble\_clf`  
  The base model or ensemble of models to be optimized.
* **param\_distributions:** `param\_dist`  
  Dictionary specifying the hyperparameters and their possible values to sample from during the search.
* **n\_iter:** `10`  
  Number of different hyperparameter combinations to try.
* **scoring:** `custom\_scorer`  
  Metric(s) used to evaluate the performance of the model during the search.
* **refit:** `'f1\_score'`  
  Metric used to select the best model after the search and refit it on the entire dataset.
* **cv:** `TimeSeriesSplit(n\_splits=10)`  
  Cross-validation strategy, using time series splits to maintain the order of observations.
* **random\_state:** `42`  
  Seed for random number generation to ensure reproducibility of results.
* **n\_jobs:** `-1`  
  Number of CPU cores to use for computation. `-1` means using all available cores.
* **verbose:** `0`  
  Controls the verbosity of the output. `0` means no output, while higher numbers provide more detailed logs.
* **error\_score:** `'raise'`  
  Determines what to do if a model fails to fit. `'raise'` means it will raise an error.

The custom\_scorer function, xgb\_early\_stopping\_score, calculates the F1 score[[4]](#footnote-4) of a model's predictions, facilitating early stopping. Early stopping is a technique where training halts if the model's performance on a validation set doesn't improve after a certain number of iterations. This prevents overfitting, ensuring the model generalizes well to unseen data. Unlike standard scorers like accuracy or F1, early stopping dynamically adjusts the training process, terminating it when optimal performance is reached, thereby improving efficiency and potentially enhancing model performance. This approach is particularly advantageous in iterative algorithms like gradient boosting, where training can be computationally expensive.

1. Ensemble Model Construction

The journey from raw data acquisition to making accurate predictions unfolds through a 3-stage process.

* 1. Data collection

The initial phase employs a Jupyter Notebook, configured for weekly data collection from the extensive repository at football-data.co.uk.

The football-data website offers match data for more than 20 seasons from a wide range of competitions from a large amount of countries, all in csv format. For our analysis, we have honed in on data from the last five seasons, specifically targeting 19 European competitions, focusing on Europe’s most important competitions, supplemented with lower tier competitions where available. This deliberate choice reflects the competitive diversity of European football, offering a dynamic and multifaceted dataset ideal for predictive modeling.

Overview of the competitions used:

**England**

* Premier League (E0)
* Championship (E1)
* League 1 (E2)
* League 2 (E3)

**Scotland**

* Premier League (SC0)
* Division 1 (SC1)

**Germany**

* Bundesliga 1 (D1)
* Bundesliga 2 (D2)

**France**

* Ligue 1 (F1)
* Ligue 2 (F2)

**Italy**

* Serie A (I1)
* Serie B (I2)

**Spain**

* La Liga 1 (SP1)
* La Liga 2 (SP2)

**Belgium**

* Pro League (B1)

**Greece**

* Ethniki Katigoria (G1)

**Netherlands**

* Eredivisie (N1)

**Portugal**

* Primeira Division (P1)

**Turkey**

* Süper Lig (T1)
  1. A computer with a light bulb and various icons

     Description automatically generatedData processing

The second phase is the workhorse of the entire project, transforming raw collected data into a polished dataset ready for the training model. This stage involves preprocessing the data collected from football-data.co.uk, cleaning, organizing, and enhancing it to ensure quality and usability.

The data-processing happens in various ways: missing values are imputed, data points are normalized, categorical data is converted into numerical formats suitable for algorithms and new features are engineered uncovering hidden patterns and relationships within the data to boost predictive accuracy. Finally, the processed data is written back to a CSV file, ready to be picked up by the training engine.

In short, this phase refines raw data into a well-structured, insightful dataset, crucial for training an effective and accurate predictive model.

* 1. Training the model

With the construction of a new dataset in phase two, we are now poised to feed this data into our ensemble model, enabling it to train and make accurate predictions on football match outcomes. Unlike simpler setups that typically split data into 80% training and 20% testing, we employ a more sophisticated approach. Recognizing the value of recent match data in reflecting current team performance, we ensure that this data is included in the training set rather than the test set.

A screenshot of a computer screen

Description automatically generated

Our training process utilizes an iterative sliding window approach. This method involves a predetermined window size, currently set to roughly 35% of the total dataset. Each window is divided into a training set and a test set (80/20) and evaluated in every iteration. Subsequently, the window shifts forward by 20% and the training process is repeated. Moreover, the model retrains on data from the previous window where incorrect predictions were made, enhancing its accuracy in subsequent iterations.

To illustrate, our most recent dataset comprises approximately 15,000 matches. With a window size of 5,000 and a step size of 1,000, the sliding window advances 1,000 matches at a time.

Throughout training, the data undergoes a comprehensive pipeline ensuring meticulous preprocessing. This includes handling missing values, encoding categorical variables, scaling features, balancing classes, selecting the most relevant features, and finally fitting the chosen classifier to the prepared data. This rigorous process guarantees that our model is well-equipped to deliver precise and reliable predictions.



**ImbPipeline:** A custom pipeline (from the imbalanced-learn library) designed to handle imbalanced datasets while allowing for easy integration of various preprocessing steps and a classifier.

**'imputer', SimpleImputer(strategy='mean'):** Fills missing values in the dataset with the mean value of the corresponding feature, ensuring no missing data hinders model training.

**'target\_encoder', TargetEncoder():** Encodes categorical features using the target variable, replacing each category with a numerical value based on the mean of the target variable for that category, helping to capture the relationship between categorical features and the target.

**'scaler', StandardScaler():** Standardizes features by removing the mean and scaling to unit variance, ensuring that each feature contributes equally to the model by having similar scales.

**'min\_max\_scaler', MinMaxScaler():** Scales features to a specified range, typically [0, 1], which can be beneficial for models sensitive to the scale of input features, like neural networks.

**'smote', SMOTE(random\_state=random\_state, k\_neighbors=2):** Synthetic Minority Over-sampling Technique (SMOTE) is used to balance the dataset by generating synthetic samples for the minority class, improving model performance on imbalanced datasets.

**'select', SelectKBest(chi2, k=fts):** Selects the top k features based on the Chi-squared statistical test, reducing dimensionality and potentially improving model performance by focusing on the most relevant features. Currently set to 14.

**'clf', clone(base\_estimator):** The classifier to be trained, cloned from a base estimator to ensure a fresh, untrained model instance is used for each training run, preserving the integrity of the original estimator settings.

* 1. Ensemble Techniques
     1. Voting

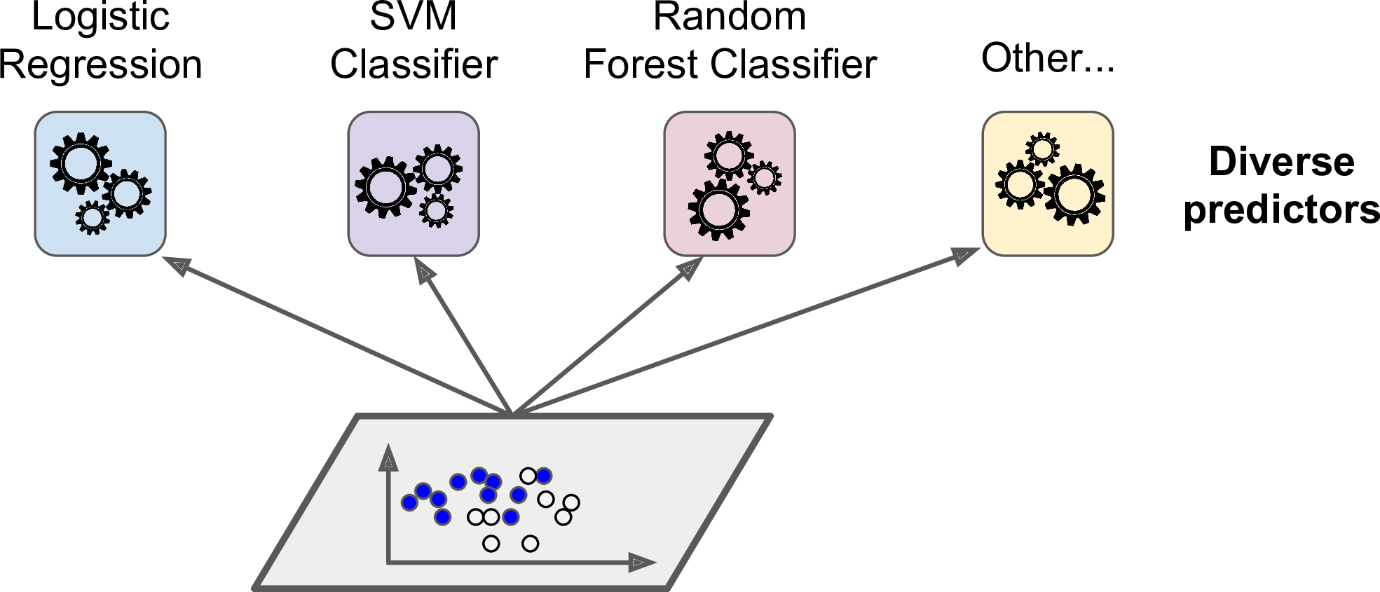
Voting combines the predictions from multiple models by averaging (for regression) or by majority vote (for classification).

**Where Voting is used:**

The VotingClassifier is created to combine predictions from multiple classifiers:



**Analysis:**  
The VotingClassifier combines multiple models and uses a soft voting mechanism, meaning it averages the probabilities predicted by each classifier.



* + 1. Stacking

Stacking refers to combining multiple models via a meta-model that makes the final prediction based on the outputs of the base models.

**Where Stacking is used:**

A StackingClassifier is not explicitly used in the provided code. While the VotingClassifier is mentioned, it doesn't implement a typical stacking approach.

**Analysis:**  
No explicit stacking is implemented in the code.

A diagram of a model

Description automatically generated

* + 1. Bagging

Bagging (Bootstrap Aggregating) involves training multiple models on different random subsets of the training data and then combining their predictions.

**Where Bagging is used:**



**Analyis:**

The RandomForestClassifier internally uses bagging by training multiple decision trees on different random subsets of the data and averaging their predictions.

A diagram of steps to a training

Description automatically generated

* + 1. Boosting

Boosting involves training multiple models sequentially, where each model tries to correct the errors of the previous ones.

**Where Boosting is used:**



**Analysis:**

The GradientBoostingClassifier, AdaBoostClassifier, XGBClassifier, and LGBMClassifier are all boosting algorithms. They train multiple models sequentially, where each new model focuses on the errors made by the previous ones.

A diagram of a process

Description automatically generated

1. Model Evaluation
   1. Evaluation Metrics
      1. A white background with black text

         Description automatically generatedAccuracy

https://www.evidentlyai.com/classification-metrics

In the intricate domain of machine learning, accuracy stands as a pivotal performance metric, encapsulating the proportion of correctly predicted outcomes to the total number of predictions rendered. This metric, while seemingly straightforward, wields considerable power in the initial evaluation of a model's efficacy. For instance, envision a model designed to forecast the results of football matches; accuracy, in this context, would signify the percentage of matches where the model's prediction (whether a team wins, loses, or draws) aligns perfectly with the actual result. However, the utility of accuracy can be deceptive, particularly in scenarios involving imbalanced datasets where certain outcomes disproportionately dominate. Thus, while accuracy provides a foundational understanding, it necessitates supplementary metrics to present a holistic view of a model's true performance.

A white rectangular object with red text

Description automatically generated

2https://www.evidentlyai.com/classification-metrics

* + 1. Precision

Precision in machine learning is about how often the model is correct when it predicts a specific outcome. For example, if a model predicts that a football team will win, precision tells us how many of those win predictions are actually correct. High precision means the model rarely mistakes a loss or draw for a win, making it very accurate in its win predictions. This is especially important in predicting football match outcomes where getting the exact result right is crucial for accuracy.

* + 1. A white rectangular object with red text

       Description automatically generatedRecall

3https://www.evidentlyai.com/classification-metrics

Recall, in machine learning, is a metric that measures the ability of a model to correctly identify all relevant instances in a dataset. It is defined as the ratio of true positive predictions to the sum of true positive and false negative predictions. High recall means that the model is able to find most of the relevant results. In football terms, if your model predicts the outcomes of matches, recall would indicate how well it correctly identifies all the matches where a team is predicted to win and they actually win.

* + 1. F1

In machine learning, the F1 score is a crucial metric for evaluating model performance, particularly in classification tasks. It combines precision and recall into a single measure, balancing the trade-off between false positives and false negatives. Specifically, precision measures the accuracy of positive predictions, while recall assesses the model's ability to identify all positive instances. The F1 score is especially valuable in contexts where the cost of false positives and false negatives are significantly different, providing a more nuanced assessment of model effectiveness.

A screenshot of a computer

Description automatically generated

4Classification report on the performance on the testset of my ensemble model

* 1. A graph with blue squares and numbers

     Description automatically generatedConfusion Matrix

A confusion matrix is a fundamental tool in machine learning used to evaluate the performance of classification algorithms. It presents the outcomes in a matrix format, allowing one to see the true positives, true negatives, false positives, and false negatives. By examining the matrix, you can understand not just the accuracy of your model, but also the types of errors it makes, providing deep insights into its strengths and weaknesses.

In the realm of machine learning, the confusion matrix is an indispensable tool for assessing the performance of classification models. By presenting predictions in a matrix format, it delineates the number of true positives, true negatives, false positives, and false negatives. This visualization allows one to interpret not just the model's overall accuracy, but also the specific types of errors, offering a comprehensive understanding of model performance nuances.

**Interpretation of each part of the matrix:**

0 = Home team won

1 = Draw

2 = Away team won

**True Positives (Correct Predictions):**

853 matches where the model correctly predicted the home team would win (0, 0).

219 matches where the model correctly predicted a draw (1, 1).

633 matches where the model correctly predicted the away team would win (2, 2).

**False Positives and False Negatives (Incorrect Predictions):**

130 matches where the model wrongly predicted a home win, but it was a draw (0, 1).

230 matches where the model wrongly predicted a home win, but it was an away win (0, 2).

218 matches where the model wrongly predicted a draw, but the home team won (1, 0).

240 matches where the model wrongly predicted a draw, but the away team won (1, 2).

122 matches where the model wrongly predicted an away win, but the home team won (2, 0).

76 matches where the model wrongly predicted an away win, but it was a draw (2, 1).

1. A machine learning approach that combines multiple individual models to improve predictive performance and robustness compared to any single model alone. [↑](#footnote-ref-1)
2. The practice of streamlining and automating the deployment, monitoring, and management of machine learning models in production to enhance reliability and efficiency. [↑](#footnote-ref-2)
3. Home/Away team shots, Home/Away team shots on target [↑](#footnote-ref-3)
4. The F1 score is a metric that combines precision and recall to provide a single measure of a model's accuracy, particularly useful for imbalanced datasets. [↑](#footnote-ref-4)