Real-time Football Statistics Provider and Machine Learning Algorithm to Predict Future Premier League Matches

Ryan Vaughan

September 2023

MSc Computing

School of Computer Science and Informatics

Cardiff University

Supervisor: Dr. Oktay Karakus

GitHub: https://github.com/RyanOliverV/VornMetrics

Abstract

This project aimed to develop a comprehensive football statistics platform that offers a blend of historical and real-time data while utilizing machine learning to predict future match outcomes. The project was guided by five key objectives: designing a user-friendly interface, curating accurate and comprehensive data, implementing real-time match updates, researching, and employing machine learning algorithms, and evaluating the predictive performance of the chosen algorithm. Employing a technology stack that includes Django for the back end and React for the front end, the project successfully meets and in some respects surpasses its objectives.

The platform's interface offers an intuitive user experience, meeting the first objective. It also successfully curates and integrates reliable football data, fulfilling the second objective. Real-time updates add a layer of dynamism, satisfying the third objective. A Random Forest Classifier algorithm was employed for its versatility, fulfilling the fourth and fifth objectives by demonstrating predictive capabilities statistically superior to random guessing.

However, the project also identifies areas for improvement, including the need for refining the predictive accuracy of the machine learning model, addressing scalability issues of the back end architecture, and mitigating latency issues for certain API-sourced data. These limitations notwithstanding, the project contributes significantly to the field of sports analytics by demonstrating the integration of traditional statistical methods with machine learning for predictive analysis in football. It also has practical implications, offering a versatile platform for various stakeholders such as fans, analysts, sports betting firms, and football clubs.

Acknowledgements

I'd like to thank Dr. Oktay Karakus, my supervisor, for guiding me throughout this project, especially with his guidance in creating a clear roadmap for the development of the website and machine learning algorithm to ensure I was able to complete every aspect.

I am deeply thankful to all the professors and lecturers I had the privilege to learn from this year. Their insightful teachings enriched my understanding and equipped me with knowledge that went beyond the confines of textbooks.

My peers—friends and classmates—deserve a special mention. Their technical insights and shared experiences played a pivotal role in shaping my time at Cardiff University.

1. Introduction

In the contemporary landscape of sports and data-driven decision-making, the world of football has undergone a remarkable transformation. With the advent of technology and the explosion of data availability, enthusiasts, analysts, and professionals have been presented with an unprecedented opportunity to unravel the hidden patterns within the beautiful game. This dissertation unveils the journey undertaken to construct a comprehensive football statistics website, which not only delivers historical and real-time insights to users but also employs cutting-edge machine learning algorithms to forecast the outcomes of future matches.

Football is not merely a sport but a cultural phenomenon that invokes emotions, camaraderie, and heated discussions. Supporters invest their time, emotions, and energy into following their favourite teams and players. However, basic scores and highlights no longer satiate their curiosity in a world driven by instant gratification and a thirst for deeper insights. The motivation to build this platform arises from the aspiration to elevate fan engagement – empowering enthusiasts with the tools to explore the sport's historical nuances and live developments, enhancing their understanding and emotional connection.

This project's scope extends beyond mere aggregation of historical and real-time football statistics. It aims to construct a user-friendly website that acts as a hub for enthusiasts seeking both retrospective analyses and future projections of football matches. By offering comprehensive statistics and insights, the website will cater to fans, analysts, and stakeholders eager to delve deeper into the intricate dynamics of the sport. The centrepiece of this project, however, is developing and implementing a machine learning algorithm that harnesses the power of predictive modelling to forecast match outcomes.

The significance of this endeavour is twofold. First, it responds to the growing demand for accurate and data-driven insights within the football community. The availability of real-time data, coupled with machine learning capabilities, empowers users to make informed decisions regarding their favourite teams, players, and upcoming matches. Second, this project contributes to the evolving field of sports analytics by showcasing how technology, data, and advanced algorithms can collaborate to refine the art of predicting football outcomes.

In the following sections of this dissertation, we will delve into the technical details of the website's construction, data acquisition and processing, the architecture of the predictive machine learning algorithm, and the validation of its predictions against real-world outcomes. By the conclusion of this dissertation, it is anticipated that readers will gain a comprehensive understanding of the intricacies involved in creating a football statistics platform and the potential of machine learning to enhance match predictions.

2. Aims and Objectives

This project aims to develop a comprehensive football statistics website that offers historical and real-time insights to users while incorporating a machine learning algorithm to predict future match outcomes.

The following objectives are identified:

1. Design and implement a user-friendly website that provides a seamless and intuitive interface for users to access a wide range of historical and real-time football statistics.

2. Curate and integrate a vast array of football data from accurate sources, ensuring accuracy, consistency, and relevance. This includes match results, player statistics, team performance metrics, and contextual data.

3. Implement mechanisms to fetch and display real-time match updates, enabling users to stay connected with ongoing games and events as they unfold.

4. Research and employ machine learning techniques to create a predictive algorithm capable of forecasting future match outcomes based solely on historical data. This involves feature engineering, model selection, and training.

5. Evaluate the accuracy and reliability of the machine learning algorithm's match predictions by assessing its performance on historical matches with known outcomes.

3. Background Material

In an era characterized by information and technology, the synergy between sports and data has catalysed the emergence of innovative solutions that augment fan experiences, enhance performance analysis, and even foresee game outcomes. This chapter delves into the existing corpus of literature, commercial products, and research endeavours that contextualize the present project within the sphere of football data analytics and predictive modelling. Through a critical evaluation of existing work, this chapter seeks to underscore the necessity for a platform that amalgamates historical and real-time football statistics, while spotlighting the unique contributions and methodologies that this project brings to the forefront.

3.1 The Evolution of Football Statistics Platforms

Over the past decade, the convergence of sports and data has spearheaded a remarkable transformation in football analysis. Advanced tracking technologies and data collection methodologies have unfurled a realm of insights that was previously inaccessible. At the vanguard of this revolution are commercial platforms like FBref and LiveScore, which have revolutionized how football enthusiasts and analysts engage with the sport. These platforms adeptly intertwine historical and real-time football statistics, delivering intricate visualisations, player metrics, and match insights tailored to an increasingly data-savvy audience.

For example, FBref has positioned itself as an industry leader by offering a comprehensive database of football metrics, encompassing everything from player movements and passes to shots on goal and defensive tactics. Platforms of this calibre enable users to dissect matches with unprecedented granularity, highlighting player contributions, tactical nuances, and game-changing moments. Similarly, LiveScore remains committed to providing up-to-the-minute match updates, allowing fans to keep pace with unfolding events.

Despite these commendable advancements, a fundamental gap endures – the seamless integration of historical and real-time data. While individual platforms excel in conferring distinct insights, the holistic narrative of football's evolution often remains fragmented. Enthusiasts find themselves toggling between historical datasets and real-time updates, which undermines their grasp of the continuous thread that weaves through the sport's journey.

This project emerges as a response to this critical gap, aspiring to bridge the chasm between historical and real-time football statistics. By constructing a unified platform that seamlessly amalgamates these dimensions, the project envisions a dynamic arena where fans can traverse the past and present of the game hand in hand. This approach fosters a more comprehensive comprehension of football's evolution, enabling users to trace the trajectory of players and teams while remaining attuned to the unfolding drama of ongoing matches.

3.2 Predictive Modelling in Sports

Predictive modelling has emerged as a riveting avenue within sports analytics, tantalizingly promising the foresight of game outcomes before they unfold. In the realm of sports, football takes centre stage as one of the most popular and statistically enriched disciplines, making it a fertile ground for such endeavours. Research has notably showcased the potential of machine learning algorithms in prognosticating football match results.

For instance, the study titled "The Effect of Weather on Football Results: An Approach Using Machine Learning Techniques" (Iskandaryan et al., 2020) stands as a significant exploration into predicting football match outcomes through the prism of machine learning. This study harnessed advanced techniques like Support Vector Machines, Random Forests, Extra-Trees, and k-Nearest Neighbours, complemented by an extensive array of features. These features extended beyond conventional team performance metrics to encompass intricate weather conditions, aimed at untangling the intricate fabric of football dynamics. The findings underscored the substantial influence of weather conditions on match outcomes. The interplay of diverse weather variables such as temperature, humidity, wind speed, and precipitation emerged as pivotal determinants, intricately interwoven with team performance factors. This intricate interplay showcased the algorithms' prowess in furnishing notably accurate predictions by meticulously considering a myriad of factors.

Similarly, the research documented in "Forecasting Football Results, and the Efficiency of Fixed-odds Betting" (Goddard & Asimakopoulos, 2004) delved into the efficacy of bookmakers' odds as predictors of match outcomes. Leveraging bookmaker odds as a repository of underlying probabilities, the authors sought to evaluate the accuracy of these odds compared to advanced machine-learning models. The study transcended the surface by integrating factors such as odds fluctuations and inferred team capabilities, thereby enriching the comprehension of the intricate dynamics underpinning predictions. The findings demonstrated that while bookmakers' odds wielded predictive power, the introduced machine-learning models exhibited comparable forecasting performance. Interestingly, the study's regression-based tests highlighted the machine-learning models' capacity to capture information about match outcomes beyond the purview of bookmakers' odds, suggesting potential inefficiencies in the odds.

However, a notable commonality threads through these research papers – the incorporation of external factors, ranging from player injuries to team dynamics and betting odds. While these factors undeniably influence match outcomes, they also convolute the extraction of pure historical patterns. In contrast, this project adheres to a more focused trajectory by exclusively leveraging historical data. This approach aims to decipher the intrinsic rhythms and trends nestled within football's past, unveiling the essence of the sport's evolution. The study titled "Predicting Football Scores Using Machine Learning Techniques" (Hucaljuk & Rakipović, 2011) discovered that basic feature sets, equipped with fewer variables, often yielded superior prediction results compared to the more intricate feature sets meticulously constructed by field experts.

In essence, the predictive modelling landscape in football boasts richness and diversity, spotlighting the latent potential of machine learning in prophesying match outcomes. Nonetheless, the inclusion of external factors in prior studies introduces complexities that blur the line between historical patterns and contemporary influences. This project sets itself apart by embracing the challenge of exclusively employing historical data, thereby presenting a unique perspective on the innate dynamics that mould football outcomes.

3.3 User Experience and Design in Sports Platforms

The digital landscape has dramatically altered the way fans engage with sports. Beyond simply providing data and statistics, the design and user experience of a sports analytics

platform can profoundly impact its effectiveness and popularity. A successful platform must cater to a wide range of users, from casual fans to hardcore enthusiasts and professionals. The literature in this area spans from usability studies to the psychology of user engagement, all converging on several key factors that constitute a successful user experience.

According to a study by Orlova in 2016, effective usability is a cornerstone for any successful digital platform. Usability focuses on making the user's interaction as simple and efficient as possible. Good usability involves a logical flow of actions or information, termed as 'user flow,' that users must follow to achieve their goals. It also highlights the importance of 'cognitive load,' or the amount of mental processing power needed to use the platform. Lower cognitive load and more straightforward user flows can enhance user satisfaction and engagement.

Moreover, an intuitive layout, underpinned by robust information architecture, enhances usability and user experience. Existing literature supports this claim, emphasizing that layout is a key element affecting not just usability but also users' emotional response to the website. A well-designed layout contributes significantly to website navigability and the ability to meet user expectations (Kincl & Štrach, 2012). Navigational aids such as tabs, sidebars, and dropdown menus are highlighted as crucial components for effective website design. According to research, navigation is a key factor that enables better recall of website structure and enhances content accessibility. Such aids can help users effortlessly traverse various types of data, from player statistics to upcoming fixtures (Sutcliffe, 2002).

Visual appeal is another important factor. According to literature in the field of human-computer interaction, the aesthetic quality of a website can significantly impact user trust and engagement (Orlova, 2016). Visual elements like typography and fonts can set the mood, while readability is influenced by factors such as font size and contrast. Colour choices can have psychological and cultural impacts, affecting how users engage with the platform. Various colour harmony schemes are suggested for creating balanced and visually appealing compositions. Furthermore, there is evidence that home pages with low visual complexity are more pleasurable and make it easier for users to find the information they need. Not only are simple designs more favourable to users but they are also better remembered (recognized) than more complex ones (Tuch et al., 2009).

User retention is highly sensitive to a web platform's performance, with particular sensitivity to issues like slow loading times and unresponsive interface elements. Such issues can quickly lead to a spike in user attrition, underscoring the imperative to optimize for both speed and responsiveness. Existing literature firmly establishes a direct correlation between fast load times and higher levels of user engagement. This relationship is shown in a study by Gehrke & Turban (1999), which identified slow website speed as the most prevalent complaint among web users. The research suggests that users are generally unwilling to endure lengthy page-load times and are likely to seek alternatives, thereby contributing to decreased retention rates.

By considering these elements, derived from a wealth of studies and literature on UX design, this project aims to construct a platform that offers not just in-depth analytics and predictive modelling, but also an engaging and user-friendly experience.

3.4 Limitations and Uniqueness of the Current Project

While existing platforms and research endeavours have propelled significant advancements in football data analytics and predictive modelling, a series of limitations persist. The absence of seamless integration between historical and real-time data obstructs enthusiasts from grasping the holistic narrative of the sport's evolution. Moreover, the integration of external factors into predictive models poses challenges in discerning genuine historical trends. This project, however, confronts these limitations by furnishing a unified platform and a predictive algorithm anchored solely in historical data.

4. Problem Statement

The purpose of this chapter is to provide a comprehensive description of the problem addressed in this study. This section will discuss the background and context of this problem, emphasizing its complexities and challenges. The chapter also aims to outline the expected benefits arising from this study.

From professional leagues to international championships, the sport commands attention from fans and various stakeholders. Over the past decade, the role of data-driven decision-making in football has substantially increased. Various platforms offer different slices of this data pie, but none provide a complete picture. FBref excels in offering detailed historical data, which helps analysts, coaches, and players themselves understand past performances and trends. On the other hand, LiveScore provides real-time updates during matches, allowing fans to stay updated with the latest developments in the games they care about.

However, the crux of the problem is the absence of a one-stop platform that combines both historical and real-time data while also offering predictive insights. Users often have to hop between platforms like FBref for historical data and LiveScore for real-time updates, which makes it difficult to get a holistic view. Moreover, while FBref's historical data is rich and extensive, it doesn't provide predictive analytics for future matches based on this data. On the flip side, real-time updates from LiveScore, though invaluable for current match status, offer little to fans and analysts looking to draw long-term insights or predictions.

The lack of a unified platform creates a fragmented user experience and constrains the depth of analysis and engagement for fans, players, and other stakeholders. This fragmentation represents a significant gap, especially in an era where data analytics could offer transformational insights into the sport.

By addressing this gap, the project aims to create a unified, comprehensive platform for football analytics that combines historical data, real-time updates, and predictive analytics. This ambitious endeavour will attempt to overcome the complexities and challenges posed by data diversity, real-time processing, predictive accuracy, and user experience design.

Through this comprehensive approach, we aim to transform how fans, analysts, and stakeholders engage with football, ultimately enriching their experience and understanding of the sport.

Complexities and Challenges

- Real-Time Processing
- The real-time nature of this project demands an architecture capable of processing large volumes of data instantaneously, adding a layer of complexity to the project.
- Predictive Analytics
- Building predictive models based on historical data adds another level of complexity. This involves not just statistical modelling but also machine learning algorithms that can adapt.
- User Experience

- The platform must cater to a wide audience, from casual fans to professional analysts. Designing an interface that is both user-friendly and feature-rich is a significant challenge.
- Legal and Ethical Considerations
- Navigating data privacy laws and securing licensing agreements for data usage are crucial and complex tasks that must be addressed.

Expected Benefits

- Comprehensive Understanding
- A unified platform would offer a more in-depth understanding of the sport, bridging the gap between different data types and offering a 360-degree view of events, players, and strategies.
- Time Efficiency
- This platform would streamline the data gathering and analysis process, saving precious time for analysts and fans.
- Data-Driven Decisions
- Teams and coaches could make more informed decisions, offering a competitive edge in a game where fine margins often determine outcomes.
- Enhanced Fan Engagement
- Fans would have a new avenue for engagement, complete with deeper analytics and real-time data, enriching their experience.
- Business Opportunities
- Creating such a unified platform opens the door for numerous commercial opportunities, from analytics as a service to innovative fan engagement solutions.

5. Approach

The central aim of this dissertation is to develop a comprehensive football statistics platform. This platform seeks to provide a unified solution, integrating historical, real-time, and predictive football data. Due to the complex nature of the project, including data diversity, real-time processing, and predictive analytics, a multifaceted approach was necessary.

Given the complexities outlined in the problem statement, the initial step was a meticulous evaluation of various frameworks and technologies to support the back end, front end, and data sources.

Chosen Approach

The methodology we eventually adopted encompasses the following key stages:

- Framework Evaluation: A comparative analysis of several back end and front end frameworks was performed, considering aspects such as scalability, ease of integration, and community support. After this rigorous evaluation, Django was chosen for the back end, and React was selected for the front end.
- API Source Evaluation: Multiple API providers, including Sports Monks, API-Football, and FootyStats API were scrutinized based on the richness of their data, latency, and reliability. Sports Monks emerged as the most comprehensive choice for both historical and real-time data.
- User Interface Design: With React's component-based architecture, a user-friendly and intuitive interface will be developed to provide an optimized user experience.
- Web Scraping Historical Matches: Techniques were used to gather a dataset specifically for developing the machine learning model. This dataset is essential for training and validating the predictive algorithms for future match outcomes.
- Predictive Algorithm Development: Machine learning techniques will be researched and employed to develop a predictive model for future match outcomes based on historical data.

5.1 Framework Evaluation

Three strong contenders for the back end framework were considered: Node.js, Flask, and Django. Each option had its merits:

- Node.js: Given my familiarity with JavaScript, Node.js was an initial strong contender. It excels in handling real-time data efficiently and is known for its fast performance.
- Flask: This Python framework offers simplicity and greater control over low-level details but lacks some of the built-in features that Django provides.
- Django: This robust Python framework comes with built-in support for serializers, is useful for data standardization, and offers a plethora of features out-of-the-box.

The choice of Django as the back end framework for this project was a carefully considered decision, dictated by multiple variables that align with the project's long-term objectives and technical requirements. One of the most pivotal considerations was the project's future requirement to integrate machine learning algorithms for predicting football match outcomes. Python stands as an industry standard for machine learning tasks, known for its robust

libraries and frameworks specifically tailored for data analytics and machine learning applications. By selecting Django, which is also Python-based, the project could capitalize on the advantages of employing a single programming language across different aspects of development. This presents an invaluable opportunity for seamless integration and code reusability, as well as simplifying the overall software architecture.

When compared to other candidate frameworks, Django emerged as the most suitable option for several reasons. Flask, another Python-based framework, was considered. However, Flask is often seen as a micro-framework that comes with "batteries not included," meaning many essential functionalities such as an admin panel, authentication, and others would need to be implemented manually or via third-party libraries. While Flask offers more flexibility, it would also have potentially increased the development time for features that come pre-built in Django.

Node JS, a JavaScript runtime environment, was another viable alternative, especially considering its asynchronous architecture and efficient handling of I/O-bound operations. However, it would require switching between JavaScript for the back end and Python for machine learning tasks. This context-switching between two languages could introduce unnecessary complexity and potential friction in an already ambitious project. Additionally, using Node would negate the benefits of employing a singular language environment for all computational tasks related to the project.

An especially salient point in favour of Django is its REST Framework (DRF), which has been instrumental in addressing the complexities of data standardization. DRF's Serializers provide a highly effective means to transform complex data types, such as query sets and model instances, into native Python data types. These can then be rendered into JSON, XML, or other content formats. This feature simplifies the challenges associated with maintaining data integrity and consistency, particularly when integrating diverse data sets from different APIs, like real-time and historical football statistics. By leveraging the capabilities of Django's Serializers, the project can achieve a level of data standardization that is crucial for accurate and reliable analytics and predictions.

Moreover, Django's robust built-in features, such as an admin panel, Object-Relational Mapping (ORM), and a wide array of built-in libraries, provided a robust starting point for rapid development. Its established track record for scalability and security also inspired confidence in its ability to support the project's ambitions to scale and handle a variety of data-related challenges effectively.

In summary, Django's Python-based architecture, comprehensive built-in features, proven scalability, security metrics and the invaluable data standardization capabilities offered by its REST Framework made it the optimal choice for the back end framework, especially when considering the project's future requirements for machine learning integration.

For the front end, three frameworks were initially considered: Angular, Vue.js, and React. The selection criteria focused on ease of use, community support, scalability, and compatibility with the back end.

- Angular: A comprehensive framework backed by Google, Angular offers two-way data binding and has a mature ecosystem. However, it has a steeper learning curve and can be seen as overly complex for some projects.
- Vue.js: Known for its simplicity and flexibility, Vue.js is easier to integrate with projects using other technologies. However, it lacks some of the robust community and corporate support that Angular and React enjoy.
- React: Developed by Facebook, React offers a virtual DOM, strong community support, and a component-based architecture that promotes the reusability of code.

The selection of React as the front end framework for this project was influenced by a confluence of factors, each contributing to the overarching goal of creating an efficient, scalable, and user-friendly platform. One of the most compelling features of React is its component-based architecture. This design philosophy facilitates code reusability, thereby significantly reducing the time and effort required for the development of future features and updates. It offers a modular approach to user interface design, making the codebase easier to manage and extend.

Furthermore, the comprehensive community support surrounding React cannot be overstated. A robust ecosystem of third-party libraries and exhaustive documentation is readily available, providing invaluable resources that accelerate the development process. This is especially crucial for a project of this scale, as it allows for the quick integration of additional functionalities and troubleshooting solutions.

In terms of compatibility, React was found to be remarkably well-suited for integration with the Django REST Framework, which was chosen for the back end. The JSON-based data structures returned by Django could be seamlessly manipulated and displayed using React, simplifying the complexities associated with data fetching and state management on the front end. This harmonious interplay between the front end and back end technologies ensures smooth data flow and augments the overall user experience.

Moreover, React's nature as a Single Page Application (SPA) framework offers a distinct advantage, particularly well-suited to this project. SPAs load a single HTML page and dynamically update content as the user interacts with the app, thereby offering a smoother user experience with quicker load times and transitions. This is crucial for a platform that aims to offer real-time data updates, as SPAs can handle large amounts of asynchronous data fetching without requiring a full page reload. This capability aligns impeccably with the project’s objectives of providing real-time match updates and predictive analytics, delivering a seamless and engaging user experience.

Lastly, scalability was a key consideration, given the project's ambition to grow in both features and user base over time. React's performance metrics, particularly its efficient handling of complex states and dynamic content, made it an ideal choice. It has the capability to scale smoothly, ensuring the platform remains agile and responsive even as it evolves to meet the needs of an expanding audience.

5.2 API Source Evaluation

In the pursuit of constructing a unified platform capable of providing both historical and real-time football statistics, one of the first and foremost tasks was to evaluate potential API sources for data acquisition. This task was pivotal, as the quality, richness, and reliability of data would fundamentally shape the capabilities and effectiveness of the entire platform. Several API providers were put under scrutiny, including Sports Monks, API-Football, and FootyStats API.

Each of these APIs was evaluated based on a set of critical criteria. The richness of the data offered was among the first aspects to be considered. The selected API needed to provide not just basic statistics like match outcomes and player metrics, but also delve into data such as player positions, attacking and defensive phases, and real-time, up-to-date scheduling information. Additionally, latency was another major concern, especially for real-time match updates. A lag in data updates could compromise the user experience, making the platform less appealing for those seeking live match data. Finally, the reliability of each API—its uptime, rate limits, and historical data accuracy—was rigorously tested.

After conducting testing various APIs within the Django API Views environment, Sports Monks emerged as the most well-suited API for this project. Not only did it offer a diverse and extensive range of both historical and real-time data, but it also met the requisite latency standards for real-time data acquisition. The platform could therefore offer a full spectrum of insights from pre-match analyses to live match updates and post-match statistics. The latency in Sports Monks' real-time data updates was within acceptable limits, making it a robust solution for real-time data acquisition. Moreover, Sports Monks proved to be highly reliable in terms of data accuracy and availability, an essential attribute for a project of this magnitude where the integrity of data is of utmost importance.

A notable advantage of Sports Monks was its comprehensive documentation and guides, which facilitated a more streamlined integration process. This was especially beneficial in accelerating the project's development cycle, as it allowed for a better understanding of the API's capabilities and how best to harness them for the platform's objectives.

Selecting Sports Monks as the API source provided a strong foundation upon which the rest of the platform’s functionalities could be reliably built. The richness of the data offered could directly contribute to the machine learning algorithms employed for predictive analytics, while its low-latency real-time data capabilities could ensure a responsive and engaging user experience. Therefore, the careful evaluation and eventual selection of Sports Monks as the API source were instrumental steps in aligning the project with its overarching aims and objectives.

5.3 User Interface Design

The approach to UI design commenced with a comprehensive comparative analysis of existing platforms in the same domain. The objective was to assess the user interface, functionality, and overall user experience offered by similar services. Rather than using traditional methods like user surveys or interviews, this evaluation acted as a proxy for understanding user expectations and industry standards. It allowed me to identify both

strengths and weaknesses in competitor platforms, informing the design decisions for my platform.

The comparative analysis revealed a notable trend across existing platforms: a tendency toward complexity and information density, which often compromised readability and intuitive navigation. Such platforms appeared to prioritize data breadth over user-focused design, thereby leading to a cluttered user interface that could potentially overwhelm or alienate users.

Contrastingly, insights from previous personal experiences with gaming statistics dashboards like Leetify demonstrated a different design approach. These dashboards were built with user-centric design principles, offering a streamlined and intuitive experience. The design philosophy behind such dashboards emphasizes the essentials, using a simple, clean layout to present data in an easily digestible format. These experiences informed the decision to adopt a similar, simplified dashboard aesthetic for the platform.

The decision to implement a simplified dashboard was, therefore, a deliberate attempt to amalgamate the best of both worlds: the comprehensive data analytics capabilities observed in competitor platforms and the user-focused design evident in gaming statistics dashboards. The goal was to present complex and multifaceted data in an organized, intuitive manner that a wide array of users could understand and navigate with ease.

This user-centric design choice aimed to place a premium on user experience, recognizing that the most sophisticated data analytics tools would be rendered ineffectual if users found the interface confusing or burdensome. Therefore, simplicity and ease of use became central tenets of the design approach, guiding the selection of elements and their placement on the dashboard.

By combining the insights garnered from the comparative analysis of similar football statistic platforms and personal experiences with more user-friendly dashboards, a tailored approach to UI design was formulated. This approach aspired to deliver a balanced, effective, and user-friendly interface that would distinguish the platform in a crowded marketplace.

5.4 Web Scraping Historical Matches

The initial step in web scraping for historical matches was to identify the specific data requirements that would be instrumental in training and validating the machine learning model. Parameters like match outcomes, shooting statistics, venue, and other game-specific variables were considered vital. Identifying these data points served as a guideline for what to focus on during the web scraping process.

Before beginning the actual scraping, suitable websites with reliable historical match data were identified. Ethical guidelines and legal constraints, such as compliance with the website's terms of use and robots.txt files, were carefully reviewed. The goal was to ensure that the web scraping was conducted responsibly and ethically. The decision to use FBref as the source for scraping historical matches was informed by multiple compelling factors. Firstly, FBref sources its data from Opta, a well-regarded sports analytics company that utilizes a combination of human annotation, computer vision, and AI modelling to collect

real-time sports statistics. This ensures a high degree of accuracy and reliability in the data, which is essential given its intended use for machine learning applications.

5.5 Predictive Algorithm Development

The overarching goal of the predictive algorithm is to enable accurate football match predictions based on historical data. Given the centrality of this feature to the platform, setting rigorous accuracy metrics as initial benchmarks is critical. These metrics will be used later to evaluate the effectiveness of the algorithm.

In anticipation of the data cleaning needs, the approach includes specific steps for handling missing values, removing outliers, and dealing with any potential class imbalance. Furthermore, feature selection, data normalization, and dataset partitioning into training, validation, and testing sets are planned to create a dataset optimized for machine learning.

Instead of testing multiple algorithms, an in-depth research approach will be taken to determine the most suitable algorithm for predicting match outcomes. Initial candidates—Logistic Regression, Random Forest, and Support Vector Machines (SVM)—will be researched in terms of their historical success in similar applications, computational efficiency, and compatibility with the type and scale of data used in this project. Based on this research, a Random Forest Classifier has been identified as the most fitting choice.

Once the algorithm is selected based on research, the next phase is to align its performance with predefined accuracy benchmarks. Hyperparameter tuning is planned for this stage to optimize the algorithm's performance, with the tuning methodology being informed by the research conducted and the unique characteristics of the chosen model.

The model's efficacy will be assessed by running it on previous data, to gauge its generalizability and robustness. By meticulously planning each of these stages—from initial objectives and data handling to algorithm selection based on research and subsequent validation—this approach aims to create a comprehensive and systematic roadmap for developing a predictive algorithm that aligns closely with the project's objectives and long-term vision.

6. Implementation

6.1 Setting up the Back end.

The first critical step in the implementation phase was establishing a robust development environment. A Python virtual environment was created using Python's built-in venv module. The rationale behind this was to isolate the project dependencies, ensuring that there would be no conflicts with global Python packages. This level of isolation is crucial for the long-term maintainability and scalability of the project, making it easy to replicate the environment elsewhere. I learnt this in the Fundamentals of Programming module as we were being taught about programming languages and frameworks. After creating the environment with 'python3 -m venv venv', it was activated, and its functionality was validated by confirming the absence of extraneous packages through the pip freeze command.

To further support the development process, Git was employed for version control. This would enable better tracking of changes and facilitate rollback in case of issues. A Git repository was initialized in the root directory of the project, providing a layer of version control right from the outset.

After activating the Python virtual environment, Django was installed via pip, the Python package manager. Following installation, a new Django project was initiated using the Django-admin startproject command. The Django development server was then run to ensure that the project setup was successful. The default Django welcome page served as a validation point, confirming the project had been set up correctly.

Contrary to traditional setups that might involve various models and database structures, this project did not require such components because the data was to be consumed from an external API. Instead, a new Django app named api was created to focus on data serialization and API endpoint creation. This modular approach allowed for a separation of concerns, making it easier to manage the application's various aspects. Furthermore, the name aligns well with the RESTful approach being employed, as it suggests that the app serves as an API layer.

The Django REST Framework (DRF) was integrated by installing it via pip with pip install djangorestframework. The api app and Django rest framework were then added to the INSTALLED_APPS list in Django's settings.py. With DRF in place, serializers were developed to translate data coming from the external API into a format suitable for front end consumption. These serializers acted as the middleware that translated the API data into JSON objects, facilitating easy data consumption by the front end.

6.2 Setting up the Front end.

Following the back end setup, my attention was geared towards implementing the front end, where React was chosen as the ideal library for building the user interface. To host the front end, a new application, aptly named 'front end', was initiated within the Django framework. The very first step was to install Node.js and npm, serving as the runtime environment and package manager for JavaScript, respectively. Rather than using the typical create-react-app scaffold, I opted for a more tailored approach, executing the npm install react react-dom command to specifically install React and ReactDOM for this project.

To enforce a modular and maintainable architecture, an organized directory structure was meticulously designed. Specifically, designated folders were established for React components; the folder ‘./front end/src/components’ was dedicated to housing React components, serving as the building blocks of the user interface, while ‘./front end/src/static’ was utilized for storing assets like images and stylesheets. This organizational approach adheres to best practices and enhances codebase maintainability, a crucial consideration in both academic and organizational settings.

Webpack was the next critical tool integrated into the project, serving as the module bundler. It amalgamates various assets, such as JavaScript, CSS, and image files, into bundled output files that can be easily served to the browser. Crucially, Webpack also supports features like code splitting, tree shaking, and asset management, which are indispensable for performance optimization.

One of the distinctive features of the implementation was the usage of a single webpack.config.js file for both development and production environments. Although only one configuration file was used, distinct build processes were facilitated through npm scripts defined in the package.json file.

By leveraging these scripts, I was able to initiate different build processes using the same Webpack configuration. The --mode development flag facilitated a build tailored for development activities like debugging, while the --mode production flag triggered optimizations such as code minification to improve performance in the production environment.

Then, Babel was set up and configured. Serving as a specialized JavaScript compiler, Babel's functionalities were particularly instrumental in fulfilling two key objectives. Firstly, it facilitated the transpilation of modern JavaScript (ES6 and later versions) into syntax that is readily interpretable by a broader spectrum of web browsers. Secondly, it was responsible for converting JSX—React's syntactic extension—into pure JavaScript, making it suitable for browser execution.

The configuration for Babel was systematically articulated within a babel.config.json file, allowing for centralized management and easy adjustments. Critical to the success of this endeavour was the strategic utilization of specific Babel presets: namely, @babel/preset-env and @babel/preset-react. By integrating these presets into the Babel configuration, the project was endowed with a dynamic mechanism that automatically transpiled the avant-garde JavaScript and JSX syntax into a form that boasted expansive browser compatibility.

After the setup of Webpack and Babel, the front end was further enhanced by integrating Material-UI, a popular React UI framework grounded in Google's Material Design principles. This choice was influenced by multiple factors that resonate well within both academic and organizational contexts. First and foremost, Material-UI streamlines the development process by providing a comprehensive suite of pre-styled components such as buttons, dialogue boxes, and more. This not only accelerates the developmental timeline but also ensures that no compromises are made on the aesthetic or functional aspects of the application.

Secondly, the consistency and coherence that Material-UI offers are invaluable. Its uniform design principles and guidelines ensure that the user experience remains unified across different sections and functionalities of the application. This harmonizes with the ethos of creating intuitive and user-friendly interfaces, a principle that holds weight both in academic usability studies and in real-world applications.

Thirdly, Material-UI does not limit the creative latitude of the development process. Even though it provides a plethora of pre-defined components, it is designed to be highly customizable. This feature allows the application to be tailored to meet specific organizational branding requirements or unique design elements, thereby rendering the framework both flexible and robust.

Lastly, Material-UI's inherently responsive design ensures that the application can cater to various device sizes without compromising the user experience. Given the ubiquity of mobile devices, this responsiveness is essential for broad accessibility and usability.

Once MUI was installed, a theme.js file was developed to keep the colour schemes, font sizes and types consistent throughout the website. Its primary function is to enhance the user experience by providing an adaptable visual environment through a toggleable light-dark mode. By consolidating all theming-related variables and settings into this singular file, developers achieve several objectives: consistency in design, ease of maintenance, and improved customizability. An example of the light and dark themes is shown in Appendix 1.

The theme.js file aimed to facilitate quick and efficient changes to the visual appearance of the application without requiring adjustments in multiple locations. This centralization allowed me to make global changes that are automatically propagated throughout the application, thus ensuring design coherence. Additionally, by offering an in-built mechanism to switch between light and dark modes, it significantly improves the user's ability to personalize their interface experience based on their preferences or ambient conditions.

6.3 Front end and Back end Integration

Integrating the front end and back end required meticulous planning to ensure seamless interaction between Django and React. To facilitate this integration, an initial HTML template was created to serve as the landing page. This template was routed through Django's urls.py file located in the 'front end' folder. Crucially, a Django view function was employed to return this HTML file, ensuring that it could be rendered by the Django framework and served to the client. Once served, this HTML file became the point of attachment for a JavaScript bundle that was generated by Webpack.

The JavaScript bundle included the compiled React components and other front end logic. Upon being loaded, React took control of the Document Object Model (DOM), with its main component, commonly referred to as 'App,' serving as the entry point for all front end logic. This 'App' component, in turn, utilized React Router to handle client-side routing. This setup provided a dynamic, single-page application experience, ensuring seamless navigation and interaction without requiring full-page reloads.

This process was aided by a book written by Valentino Gagliardi, titled 'Decoupled Django', which delves into the intricacies of decoupled architectures utilizing Django and the Django

REST Framework. This pivotal resource elucidated diverse approaches for cohesively integrating Django and React. In the context of this project, Django was intentionally configured to operate as an API, with the primary responsibility of loading a singular HTML template. Subsequently, React was given the reins to govern the front end, thus creating a decoupled, yet harmonious, architectural design.

6.4 API Integration

In the next phase of implementation, attention was turned towards the back end, specifically within the api Django app, to retrieve data from the SportMonks API. Django views were employed to access this third-party API, and careful data manipulation was undertaken to filter and procure the specific datasets required, such as data for particular seasons, players, and teams. The raw data retrieved was then converted into JSON format, marking the first step in preparing the data to be used on the front end. The raw datasets, initially displayed in a nested and complex structure, were then meticulously parsed, and translated into a more manageable JSON format, laying the cornerstone for subsequent front end integration.

Building upon Django's inherent capabilities, Django Rest Framework (DRF) was used for further data serialization. The DRF serializers provided a more refined control over the JSON output, allowing for tailored data structures that meet the front end's specific requirements. Custom methods were devised within the serializer to fetch precise data fields, rename them for consistency, and format them in a manner that eases front end data handling. An example of this approach is when I calculated the number of matches played, wins and losses, and on another occasion iterated through the list of teams to return the correct team with a particular ID. The ‘Fundamentals of Programming’ module was incredibly useful in preparing me for this section, providing me with the knowledge of using 'for loops' and 'if statements' in Python which I used to iterate through the JSON dictionary and return the desired outputs.

An example of this implementation is included in Appendix 2 and Appendix 3. In this example, the TeamDetail class (which is an example of the view) functions as the controller for the API endpoint, responsible for interacting with the SportMonks API, filtering the relevant data, and then serializing it. The TeamDetailSerializer class acts as the model that shapes the JSON response, and it also contains the logic for custom data processing steps. Together, they enable the back end to serve detailed and specific information about a football team, which can then be readily consumed and displayed on the front end. This structured approach makes it easier to maintain the codebase and ensures that the front end receives high-quality, relevant data optimized for user interaction.

With the back end adequately set up to serve well-structured JSON data, the focus shifted to the React-based front end. Utilizing React's native useState and useEffect hooks, asynchronous data fetching was implemented with precision. The useEffect hook was configured to trigger API calls upon the component's mounting, funnelling the inbound data into a state variable managed by useState. This state variable was then used for rendering data dynamically on the front end.

Material-UI, previously integrated for its robust UI components, played a significant role in this context. It was employed to create data presentation elements—such as tables and cards—that are not only visually appealing but also functional. Material-UI's components

were chosen for their ease of customization and built-in responsive design features, which further enhanced the data display across multiple devices. The Appendices showcase various features of the website that present the statistics sourced from the API. Appendix 4 displays the Live Score fixtures page, which highlights either upcoming or current matches with their real-time scores. Appendix 5 provides a glimpse into the detailed in-game statistics updated in real-time. The Premier League schedule and results, with the predicted outcomes, are evident in Appendix 6. Appendix 7 presents team statistics, offering both a general overview and more specific metrics related to offensive and defensive plays. Finally, Appendix 8 shares player statistics,

6.5 Web Scraping Football Match Data

The first pivotal step in constructing a machine-learning model capable of predicting future Premier League football matches was the acquisition of historical data. To this end, the Python programming language was employed alongside specific libraries tailored for web scraping and data manipulation, namely the requests and Beautiful Soup libraries.

The initial point of entry for web scraping was FBref's website, which houses current Premier League standings. Utilizing Python's requests library, an HTTP GET request was used to retrieve the webpage content. This raw HTML data was then parsed through the Beautiful Soup library, enabling more refined data manipulation in subsequent steps.

The parsed HTML content contained a table labelled 'table.stats_table' in the source code, offering a variety of team-specific data. By employing Beautiful Soup's 'find_all' method, the hyperlinks embedded within this table were methodically identified. A filtration process was then applied to retain only the hyperlinks containing the '/squads/' substring, which are indicative of pages containing comprehensive statistics for individual teams. The base URL was concatenated with these filtered, partial URLs to create the complete URLs that would serve as the endpoints for further data scraping.

Each of these complete URLs was subsequently accessed through individual HTTP GET requests. The HTML content returned was then parsed to convert tables containing match details into Data Frame objects using Pandas' 'read_html' function. These Data Frames held key attributes such as match dates, scores, and fixtures.

To elevate the depth of the dataset, the web scraping process was further developed to encompass shooting statistics. Additional HTTP GET requests were sent to fetch these shooting metrics. Pandas' merge method was then invoked to amalgamate this shooting data with the prior match details, based on the common "Date" column. This fusion resulted in a multidimensional data frame.

Recognizing the requirement for data that spanned multiple seasons, an iterative loop was constructed. This loop ranged from the 2024 season back to 2018. During each cycle of this loop, the identical web scraping process was employed to amass data on match and shooting statistics for the respective seasons. To ensure ethical web scraping practices, a sleep interval of 10 seconds was interjected between consecutive HTTP requests through Python's 'time.sleep()' method.

The culmination of the web scraping process saw the integration of the gathered data into a monolithic data frame. The column names of this Data Frame were standardized by transforming them into lowercase strings. Lastly, the enriched Data Frame was serialized into a CSV file via Pandas' ‘to_csv’ function. This CSV file now serves as a robust data foundation for the machine learning model development.

6.6 Developing the Machine Learning Model

This section explains the process of the implemented machine learning model to predict future Premier League matches. It utilized Pandas for data manipulation, Scikit-learn for model training and evaluation, and Pickle for model serialization. The model uses a Random Forest Classifier and includes various preprocessing steps like One-Hot Encoding and Ordinal Encoding.

The data source was the previously web-scraped football matches dataset. This file contained a plethora of fields including, but not limited to, team names, dates of matches, goals scored, venue, and results. Once this file was loaded into a Pandas Data Frame, the process could begin. The initial step in the data preprocessing phase involved the transformation of the result column, which contained textual descriptions of match outcomes such as "W" for wins, "D" for draws, and "L" for losses. A new column named target was introduced, where the outcomes were numerically encoded as follows:

- Loss was encoded as 0.
- Draw was encoded as 1.
- Win was encoded as 2.

After encoding, the original result column was dropped to avoid redundancy.

The data was then divided into training and validation subsets. This was performed using Scikit-learn’s train_test_split function, with 80% of the data allocated for training and the remaining 20% for validation. A random seed was set to 42 to ensure the experiment's reproducibility.

To prepare the data for machine learning, feature engineering and encoding were critical steps. Two different types of encoders were used to transform the categorical data into a numerical format:

1. **OneHotEncoder**: Employed for nominal categorical variables like **team** and **opponent** names, this encoder converts each unique value into a new categorical variable and assigns a binary value of 0 or 1. For example, teams named 'Team_A' and 'Team_B' would each receive their column, where a match involving 'Team_A' would have a '1' in the 'Team_A' column and a '0' in the 'Team_B' column.

2. **OrdinalEncoder**: This was used for ordinal categorical features such as **venue** and **captain**. Unlike OneHotEncoding, this encoder assigns a single column where each unique category is mapped to an integer.

Then, data transformation was performed through a custom pipeline that took care of several tasks in a sequence:

- Date Feature Extraction: The pipeline extracted year, month, and day from the date column to include the time factor in predictions.
- Standardization: For any continuous variables, a Z-score normalization was applied.
- Rolling Averages: In football, a team's recent performance could be indicative of its future performance. To capture this, rolling averages of features like goals scored were computed for a 5-game window.
- Encoding: The initialized encoders were applied to the relevant columns.

The Random Forest Classifier algorithm was chosen for model training, primarily due to its ability to perform well on both classification and regression tasks, and its robustness against overfitting. After training the model on the pre-processed data, the feature importances were evaluated to identify which variables contributed most to the model's predictive power.

The model was evaluated using the validation dataset. Performance was assessed through accuracy and precision. The accuracy and precision scores changed slightly each time the program was run; however, they were consistently scoring between 0.47 and 0.50 for accuracy, and between 0.40 and 0.42 for precision. These metrics provided a view of the model's effectiveness in predicting football match outcomes.

After satisfactory performance metrics were obtained, the trained model was serialized using Python's Pickle library. This allowed for the model to be saved as a .pkl file, thereby making it easy to integrate into the website.

6.7 Integrating the Machine Learning Model into the Website

The culmination of this project involved the integration of the developed machine learning model into the web application to provide users with predictions for upcoming Premier League matches. A meticulous methodology was implemented to ensure seamless integration, starting with the placement of the machine learning code within the back end architecture and extending to the delivery of prediction results on the front end. This section explains the detailed process that was followed to implement this integration.

The machine learning model was integrated into the back end infrastructure of the Django web framework, specifically within the api Django app. The code to predict the final results was incorporated into the views.py file of this app, thereby enabling the model to directly influence the application's functionality.

The Python Pickle library had been previously used to serialize the machine learning model into multiple .pkl files:

- random_forest_model.pkl: This file stores the trained Random Forest model. When predictions are needed in the future, you can load this pre-trained model, thus avoiding the need to retrain the model.
- one_hot_encoder.pkl: This file contains the OneHotEncoder object that is used for encoding team names. Keeping this file allows for the consistent application of the encoding scheme to new data sets without retraining.
- ordinal_encoder.pkl: This file contains the OrdinalEncoder object, which is used for label encoding of categorical features such as venue, captain, and referee. As with the

OneHotEncoder, saving this ensures that future encoding will match the training dataset.

- rollingDict.pkl: This dictionary captures the rolling averages and last values for various metrics. It helps in preprocessing future datasets in the same way as the training set.

These files facilitated the loading of the model into the Django application. To make predictions for future matches, a dataset containing information on upcoming fixtures was also introduced into the back end. This dataset was critical for feeding the model with the required data points.

Once the model was successfully loaded into views.py, minor adjustments were performed to convert the model's output into a dictionary format. This was an essential step, as it enabled the structured representation of prediction results, thus facilitating subsequent data serialization and transport to the front end.

Data serialization plays a critical role in ensuring that the complex data structures produced by the back end are simplified and converted into a format that can be effortlessly consumed by the front end. To that end, a specialized data serializer was implemented to standardize the model's output. This serializer transformed the dictionary of prediction results into a JSON-like format that is easily retrievable and usable within the front end architecture.

On the front end side, particularly within the 'Match Prediction' section of the web application, a sequence of carefully orchestrated operations was carried out. Initially, an HTTP request was initiated using React's fetch to retrieve and present the list of fixtures from the SportMonks API, for the upcoming Premier League season.

Once the fixture information was successfully displayed, a subsequent fetch request was executed to obtain the machine learning model's predictions, which had been processed and serialized by the back end. These prediction results were then parsed and presented side-by-side with the respective fixtures, thereby providing end-users with a comprehensive and real-time predictive analysis for upcoming Premier League matches.

7. Analysis

The overarching aim of this project was to engineer a comprehensive platform for football statistics, enriched by a machine-learning algorithm capable of predicting future match outcomes. This ambitious aim has been unequivocally actualized throughout various phases of the project. From crafting a responsive front end using React to architecting a robust back end with Django, and from judiciously integrating a reliable football API for historical data to curating additional data through web scraping, the project has comprehensively achieved its foundational aim.

- Objective 1: User-Friendly Interface
- The final product not only satisfies but exceeds the first objective, offering users a seamless and user-friendly interface to interact with a plethora of football statistics.
- Objective 2: Data Integration
- This mandated the collation and integration of accurate, consistent, and comprehensive datasets. The chosen data sources are reliable and consistent, making the platform a trusted source for football data.
- Objective 3: Real-Time Updates
- Regarding the third objective, the website is equipped with real-time match update functionalities. This feature elevates user experience by keeping users abreast of ongoing matches, thus achieving a dynamism that is critical for fan engagement.
- Objective 4: Machine Learning Algorithm
- The fourth objective entailed extensive research and deployment of machine learning techniques. Following a rigorous research phase, a Random Forest Classifier algorithm was finalized. This choice was motivated by the algorithm's versatility and robustness against overfitting.
- Objective 5: Algorithm Evaluation
- The fifth objective encompassed the empirical evaluation of the model. Though the model's accuracy and precision scores are not in the realm of perfection, they do range between 0.47 and 0.50 and 0.40 and 0.42 respectively, indicating predictive capabilities that substantially exceed random guessing.

The technology stack was deliberately chosen for its robustness and scalability. Django offers a secure and feature-rich back end development environment, while React ensures a responsive and interactive front end experience. The Random Forest Classifier further fortifies this robustness, offering excellent generalizability to new and unseen data.

The project has not only met but in many aspects exceeded its initial aim and objectives. It serves as a comprehensive, accurate, and interactive platform for football fans while demonstrating the promising inclusion of machine learning for predictive analytics. However, there is room for improvement, particularly in the machine learning model's predictive accuracy.

8. Conclusion

The project exhibits several strengths that align well with the initial objectives. The user interface and experience go beyond just being user-friendly; they offer an intuitive way for users to interact with complex data. This achievement resonates with the first project objective. Similarly, the high data integrity ensures that the platform serves as a reliable hub for football statistics, effectively satisfying the second objective. Adding real-time updates brings the platform alive, offering a much-needed dynamism that keeps the users engaged, thereby meeting the third objective. Finally, the Random Forest Classifier algorithm, selected for its versatility and ability to generalize, aligns with the fourth and fifth objectives, demonstrating a reasonable level of predictive power.

Despite these strengths, the project has several areas of weakness. The predictive accuracy of the machine learning model falls short of perfection. While the model performs statistically better than a random guess, there is still room for improvement in tuning, feature engineering, or possibly even model selection. Another deficiency relates to the scalability of the back end architecture. As the user base grows, the current back end might present limitations that need to be addressed. Finally, due to time constraints, comprehensive user acceptance testing could not be performed. This leaves a gap in understanding the broader user experience and the system's actual performance in a real-world scenario.

Additionally, the platform's performance encounters challenges in terms of load speeds, for data sourced from the SportMonks API. Specifically, the extended load times are noticeable when retrieving comprehensive player data for an entire Premier League season. This latency arises from the complex data manipulation processes required on the back end to aggregate and present this information. The delays exceed acceptable standards, thereby potentially impairing user experience and engagement.

Given more time or alternative circumstances, various enhancements could have been undertaken. These might include rigorous fine-tuning of the machine learning model or the exploration of alternative algorithms for improved predictive outcomes. Further advanced A/B testing could be employed to refine the user interface, and additional real-time features, such as live commentary, could augment user engagement. Moreover, in hindsight, an alternative approach to sourcing historical data could be considered to mitigate loading issues. Instead of relying on the SportMonks API, constructing a proprietary dataset through web scraping would offer a more scalable and efficient solution. Such a strategy would not only reduce loading times but also potentially enable the acquisition of more granular statistics.

On the academic front, this dissertation enriches the existing body of knowledge in sports analytics, particularly by integrating machine learning algorithms with traditional statistical approaches for predictive analytics in football. From a practical perspective, the platform serves a wide array of stakeholders, including football fans, analysts, and potentially even sports betting firms and football clubs. It thus demonstrates the real-world applicability of machine learning for predictive sports analytics, offering invaluable insights for various practical endeavours.

9. Reflection

Completing this project has been a transformative journey that has imparted invaluable lessons in both the realm of academic research and personal development. The project underscored the importance of time management and effective project planning. Initial timelines proved to be optimistic, especially when accounting for the challenges tied to data sourcing and machine learning model tuning. This experience has taught me to allocate a buffer for unforeseen obstacles in future projects.

Regarding the authority and availability of sources, the project acquainted me with the process of selecting reliable data sources and APIs. It made apparent that not all readily available data is of high quality or relevance, requiring rigorous verification for academic validity. It also opened my eyes to the vast array of methodological approaches available for machine learning, and how each comes with its own set of assumptions, strengths, and weaknesses. In retrospect, greater attention to preliminary research could have informed more optimized choices in methodology and data sources.

On a personal note, the project served as a magnifying glass for my strengths and weaknesses. While I found myself well-equipped in coding and data manipulation, my initial lack of expertise in machine learning algorithms exposed a key area for ongoing personal development. My strengths in project visioning and conceptual framework development were counterbalanced by a tendency to underestimate the complexities of real-world data and user experience issues, something I plan to address moving forward.

In terms of the substantive topics addressed, this endeavour provided a hands-on tutorial on the complexities of integrating traditional statistics with machine learning for predictive analytics in sports. It also offered insights into the rapidly evolving domain of real-time sports analytics and its impact on fan engagement. However, the project left some questions unanswered, such as the scalability of such a platform and the full range of factors that might influence match outcomes, representing avenues for future research.

The efficacy of the chosen Random Forest Classifier, while reasonable, could perhaps be improved with more advanced algorithms or ensemble methods. Similarly, the user interface, though intuitive, might benefit from the integration of more interactive, real-time features. This project, therefore, not only contributed to existing academic dialogue but has also set the stage for further explorations into the realms of sports analytics, machine learning, and user interface design.

In conclusion, the project has been a significant milestone in my academic journey, serving as both a reflection of my existing skills and a roadmap for areas requiring further development. The lessons learned are numerous and multifaceted, encompassing both the procedural aspects of research and deeper, subject-specific insights. These learnings are not merely academic but offer practical implications, promising to inform and enrich my future in both academic research and real-world applications.

References

Cheung, C.M.K. and Lee, M.K.O. (2005) 'The asymmetric effect of website attribute performance on satisfaction: An empirical study', Proceedings of the 38th Annual Hawaii International Conference on System Sciences [Preprint]. doi:10.1109/hicss.2005.585.

Gagliardi, V. (2021) Decoupled Django: Understand and build decoupled Django Architectures for JavaScript front ends. Berkeley, California, Apress.

Gehrke, D. and Turban, E. (1999) 'Determinants of successful website design: Relative importance and recommendations for effectiveness', Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers [Preprint]. doi:10.1109/hicss.1999.772943.

Goddard, J. and Asimakopoulos, I. (2004) 'Forecasting football results and the efficiency of fixed-odds betting', Journal of Forecasting, 23(1), pp. 51–66. doi:10.1002/for.877.

Iskandaryan, D. et al. (2020) 'The effect of weather in football results: An approach using machine learning techniques', Applied Sciences, 10(19), p. 6750. doi:10.3390/app10196750.

Kincl, T. and Štrach, P. (2012) 'Measuring website quality: Asymmetric effect of User Satisfaction', Behaviour & Information Technology, 31(7), pp. 647–657. doi:10.1080/0144929x.2010.526150.

Orlova, M. (2016) 'USER EXPERIENCE DESIGN (UX DESIGN) IN A WEBSITE DEVELOPMENT', Mikkeli University of Applied Sciences.

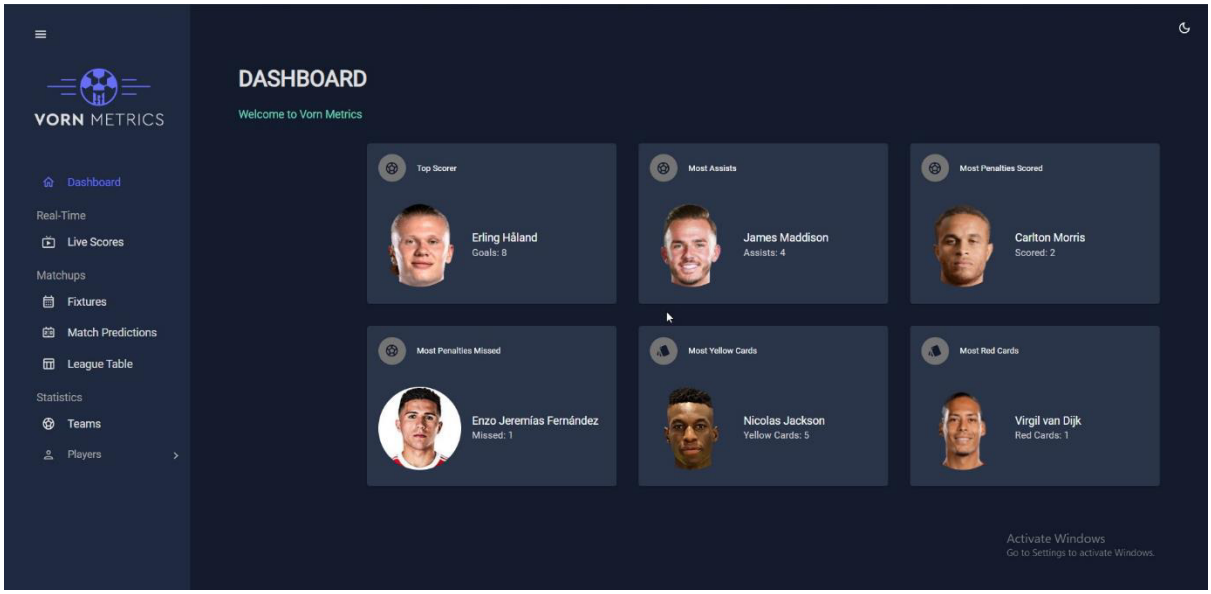
J. Hucaljuk and A. Rakipović (2011) 'Predicting football scores using machine learning techniques', Proceedings of the 34th International Convention MIPRO pp. 1623-1627.

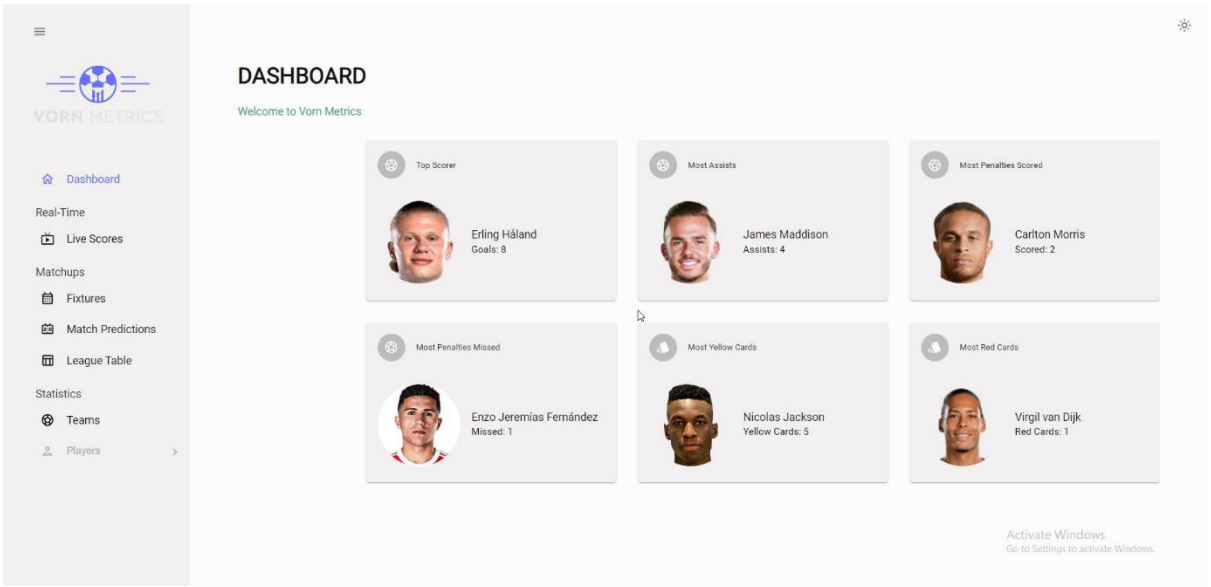
Sutcliffe, A. (2002) 'Assessing the reliability of heuristic evaluation for web site attractiveness and usability', Proceedings of the 35th Annual Hawaii International Conference on System Sciences [Preprint]. doi:10.1109/hicss.2002.994098.

Tuch, A.N. et al. (2009) 'Visual complexity of websites: Effects on users' experience, physiology, performance, and memory', International Journal of Human-Computer Studies, 67(9), pp. 703–715. doi:10.1016/j.ijhcs.2009.04.002.

Appendices

Appendix 1





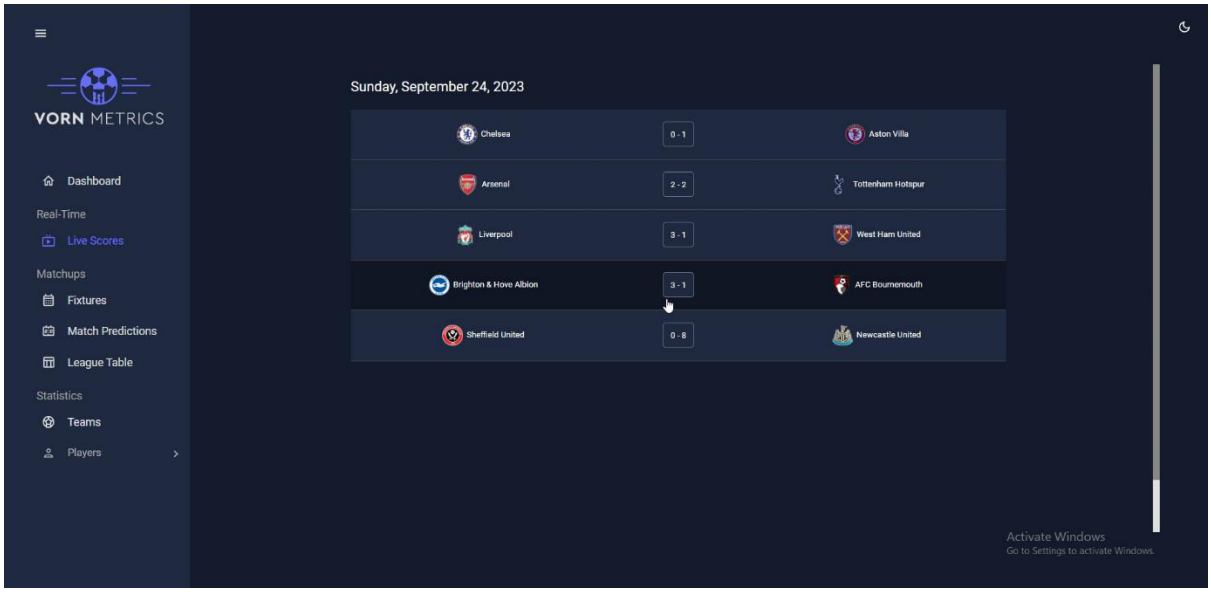
Appendix 2

```
class TeamDetail(APIView):
    def get(self, request, id):

        url = f"https://api.sportmonks.com/v3/football/teams/{id}?api_token=
      SfgFq9wDOHoDn9T5XiLZsSf2Id2rJ7lTgafxIoxOfDbwczPBrHTaQxtcmYUL&include=ve
      nue;country;coaches.coach;statistics.details.type;latest"

        response = requests.get(url)
        data = response.json()
        team = data["data"]

        # Filter statistics for the desired season_id (21207)
        season_id_to_filter = 21646
        filtered_statistics = [statistic for statistic in team["statistics"]
         if statistic["season_id"] == season_id_to_filter]

        # Update the team data with the filtered statistics
        team["statistics"] = filtered_statistics

        serializer = TeamDetailSerializer(team)

        return Response(serializer.data)
```

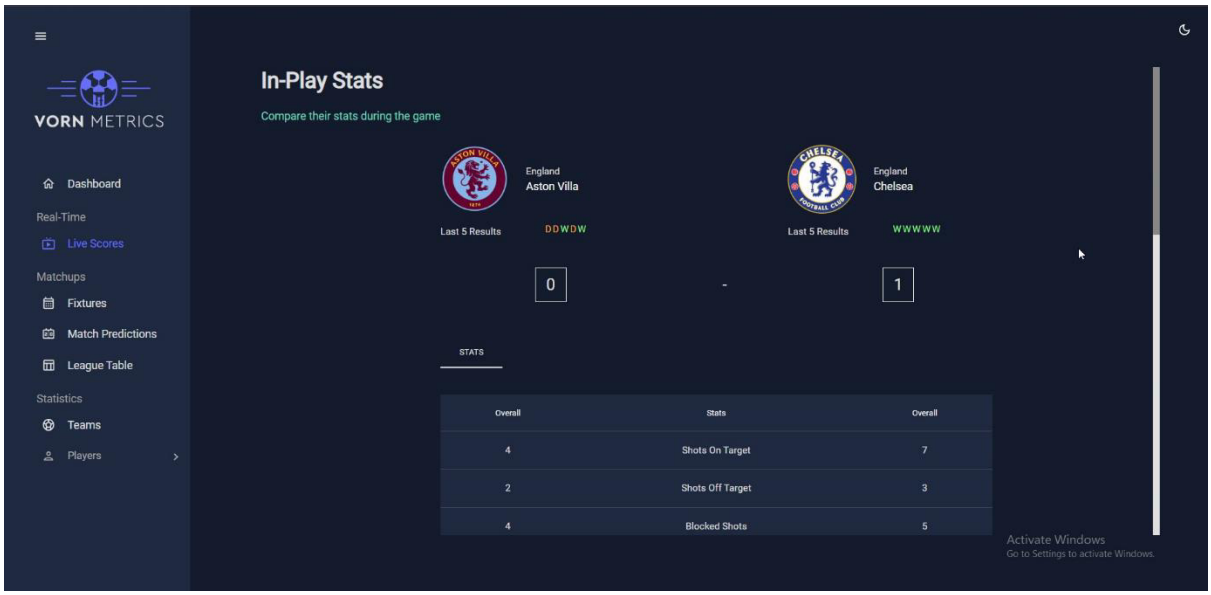
Appendix 3

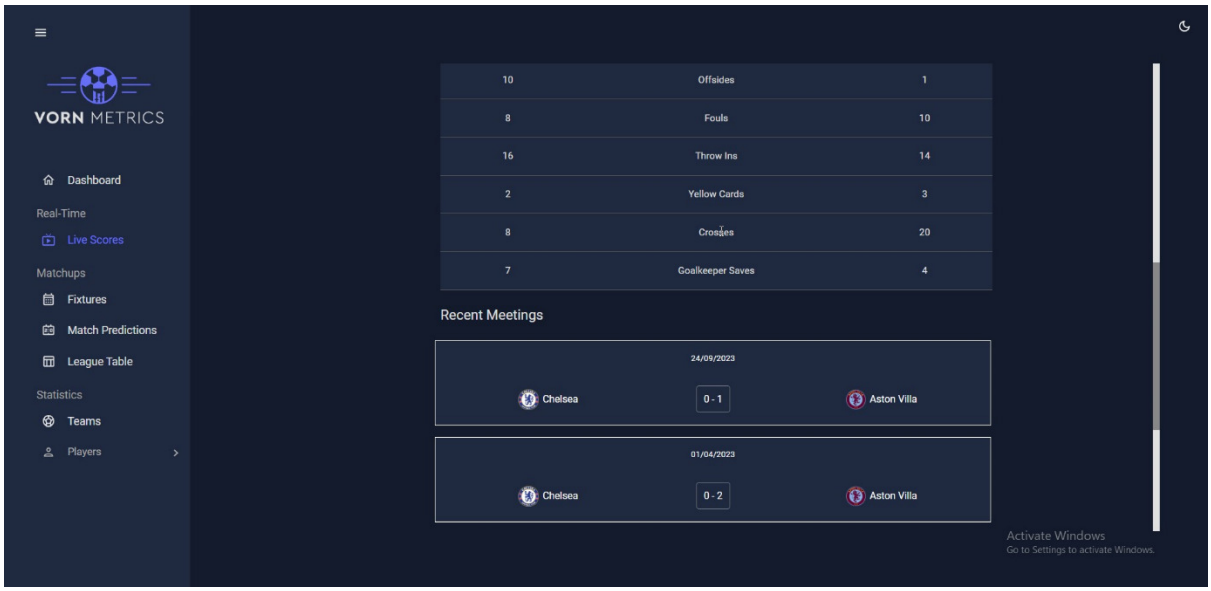
```python
class TeamDetailSerializer(serializers.Serializer):
    id = serializers.IntegerField(default=None)
    name = serializers.CharField(max_length=100, default=None)
    logo = serializers.CharField(
        source='image_path', max_length=100, default=None)
    founded = serializers.CharField(max_length=100, default=None)
    country = serializers.CharField(
        source='country.name', max_length=100, default=None)
    stadium_name = serializers.CharField(
        source='venue.name', max_length=100, default=None)
    matches_played = serializers.SerializerMethodField()
    wins = serializers.SerializerMethodField()
    draws = serializers.SerializerMethodField()
    losses = serializers.SerializerMethodField()

    def get_matches_played(self, instance):
        wins = self.get_wins(instance) or 0
        draws = self.get_draws(instance) or 0
        losses = self.get_losses(instance) or 0

        matches_played = wins + draws + losses
        return matches_played

    def get_wins(self, instance):
        statistics = instance.get('statistics', [])
        for stat in statistics:
            for detail in stat.get('details', []):
                if detail['type']['code'] == 'team-wins':
                    return detail['value']['all']['count']

    def get_draws(self, instance):
        statistics = instance.get('statistics', [])
        for stat in statistics:
            for detail in stat.get('details', []):
                if detail['type']['code'] == 'team-draws':
                    return detail['value']['all']['count']

    def get_losses(self, instance):
        statistics = instance.get('statistics', [])
        for stat in statistics:
            for detail in stat.get('details', []):
                if detail['type']['code'] == 'team-lost':
                    return detail['value']['all']['count']
```

Appendix 4

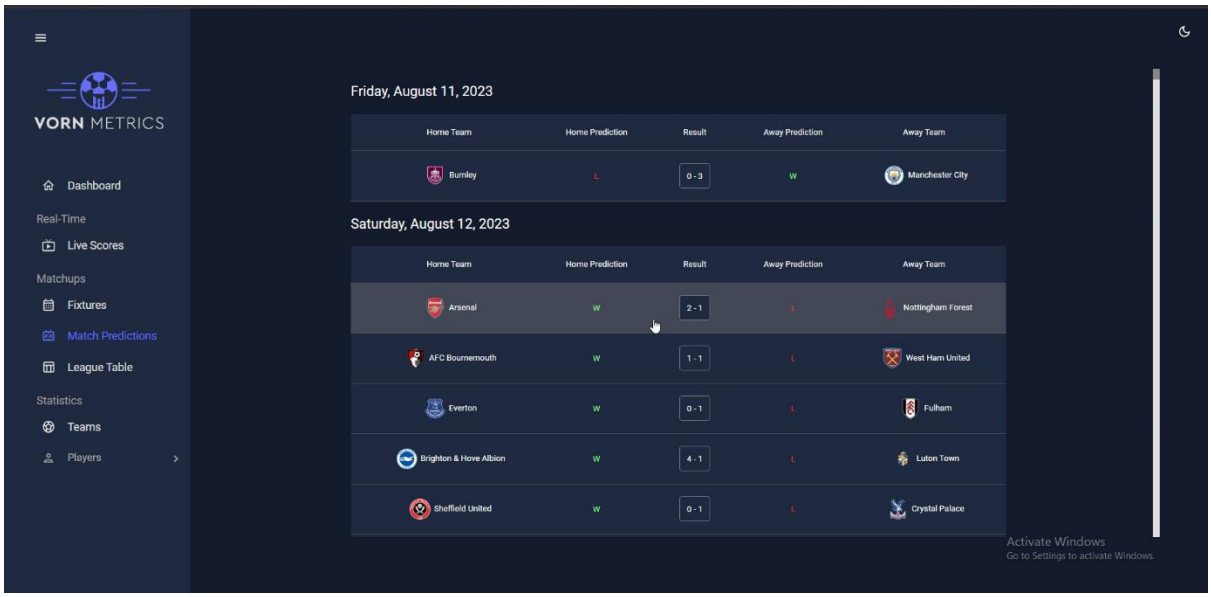

VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
Sunday, September 24, 2023
Chelsea
0 - 1
Aston Villa
Arsenal
2 - 2
Tottenham Hotspur
Liverpool
3 - 1
West Ham United
Brighton & Hove Albion
3 - 1
AFC Bournemouth
Sheffield United
0 - 8
Newcastle United
Activate Windows
Go to Settings to activate Windows.

Appendix 5

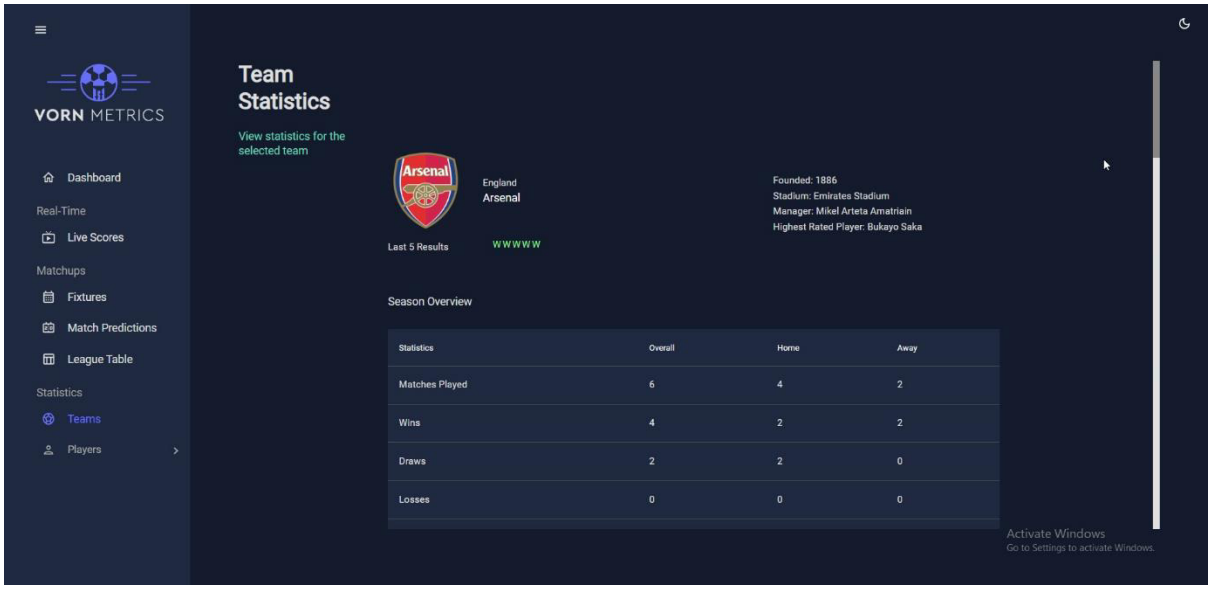

VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
In-Play Stats
Compare their stats during the game
England
Aston Villa
Last 5 Results
DDWDW
England
Chelsea
Last 5 Results
WWWWW
0
-
1
STATS
Overall
Stats
Overall
4
Shots On Target
7
2
Shots Off Target
3
4
Blocked Shots
5
Activate Windows
Go to Settings to activate Windows.

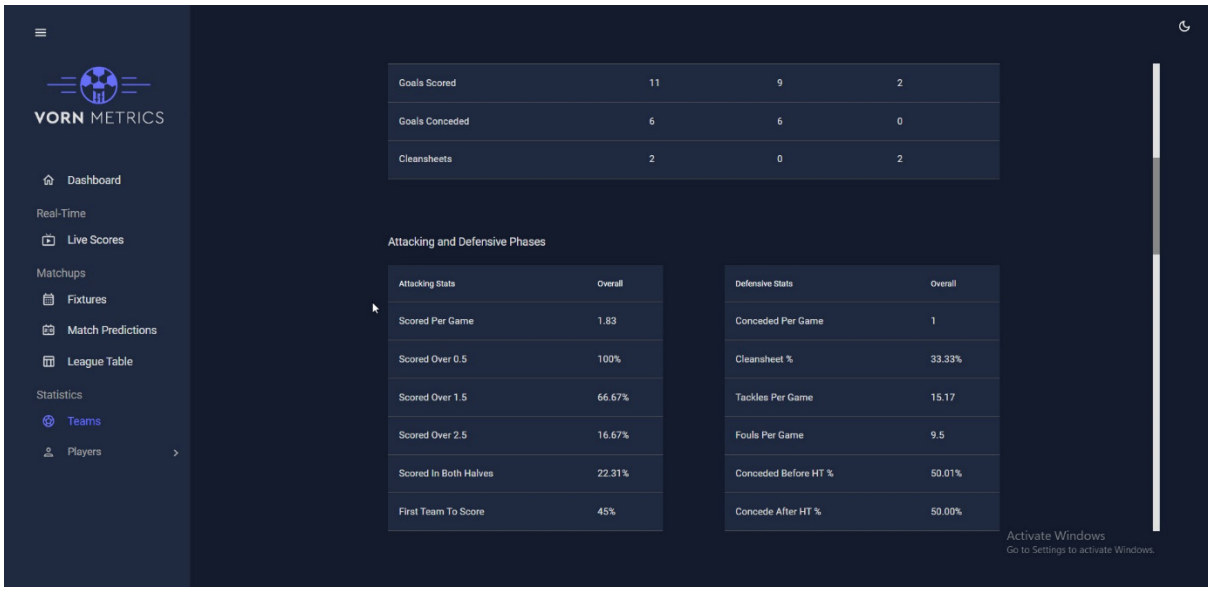

VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
10
Offsides
1
8
Fouls
10
16
Throw Ins
14
2
Yellow Cards
3
8
Crosses
20
7
Goalkeeper Saves
4
Recent Meetings
24/09/2023
Chelsea
0 - 1
Aston Villa
01/04/2023
Chelsea
0 - 2
Aston Villa
Activate Windows
Go to Settings to activate Windows.

Appendix 6

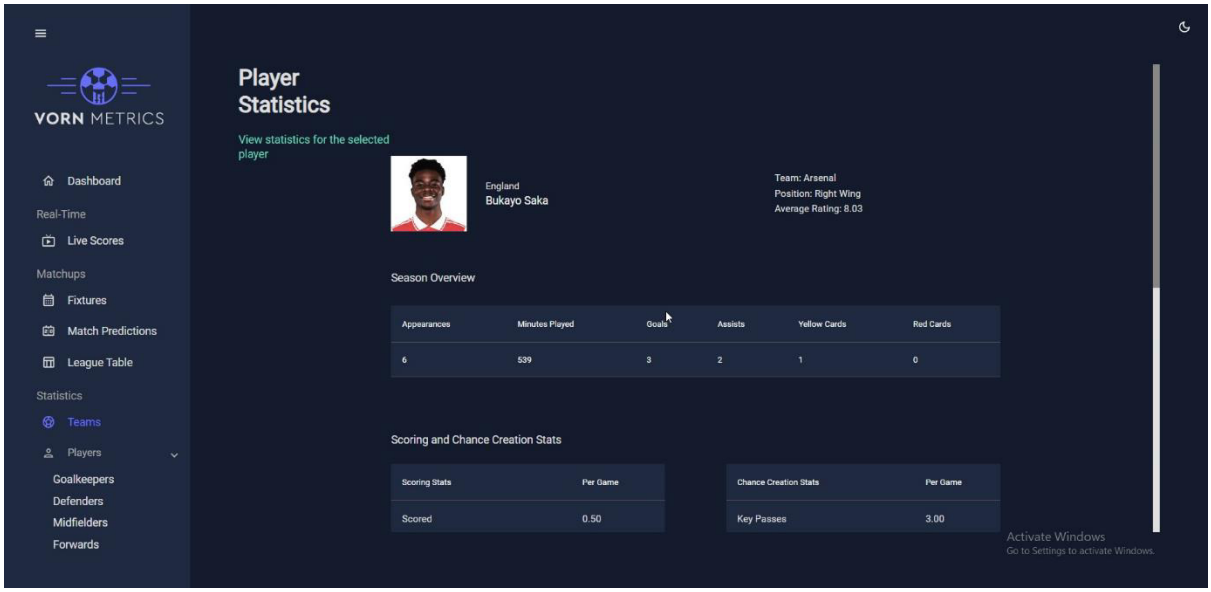

VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
Friday, August 11, 2023
Home Team
Home Prediction
Result
Away Prediction
Away Team
Burnley
L
0 - 3
W
Manchester City
Saturday, August 12, 2023
Home Team
Home Prediction
Result
Away Prediction
Away Team
Arsenal
W
2 - 1
L
Nottingham Forest
AFC Bournemouth
W
1 - 1
L
West Ham United
Everton
W
0 - 1
L
Fulham
Brighton & Hove Albion
W
4 - 1
L
Luton Town
Sheffield United
W
0 - 1
L
Crystal Palace
Activate Windows
Go to Settings to activate Windows.

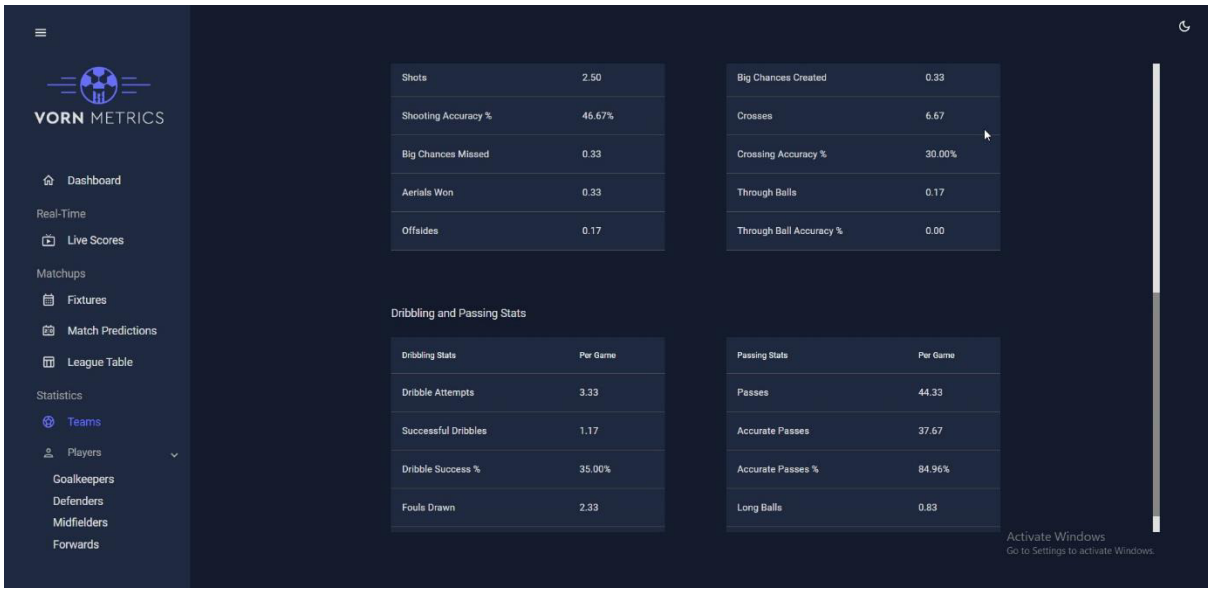
Appendix 7


VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
Team Statistics
View statistics for the selected team
England
Arsenal
Last 5 Results
WWWWW
Founded: 1886
Stadium: Emirates Stadium
Manager: Mikel Arteta Amatriain
Highest Rated Player: Bukayo Saka
Season Overview
Statistics | Overall | Home | Away
Matches Played | 6 | 4 | 2
Wins | 4 | 2 | 2
Draws | 2 | 2 | 0
Losses | 0 | 0 | 0
Activate Windows
Go to Settings to activate Windows.


VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
Goals Scored | 11 | 9 | 2
Goals Conceded | 6 | 6 | 0
Cleansheets | 2 | 0 | 2
Attacking and Defensive Phases
Attacking Stats | Overall
Scored Per Game | 1.83
Scored Over 0.5 | 100%
Scored Over 1.5 | 66.67%
Scored Over 2.5 | 16.67%
Scored In Both Halves | 22.31%
First Team To Score | 45%
Defensive Stats | Overall
Conceded Per Game | 1
Cleansheet % | 33.33%
Tackles Per Game | 15.17
Fouls Per Game | 9.5
Conceded Before HT % | 50.01%
Concede After HT % | 50.00%
Activate Windows
Go to Settings to activate Windows.

Appendix 8


VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
Goalkeepers
Defenders
Midfielders
Forwards
Player Statistics
View statistics for the selected player
England
Bukayo Saka
Team: Arsenal
Position: Right Wing
Average Rating: 8.03
Season Overview
Appearances
Minutes Played
Goals
Assists
Yellow Cards
Red Cards
6
539
3
2
1
0
Scoring and Chance Creation Stats
Scoring Stats
Per Game
Scored
0.50
Chance Creation Stats
Per Game
Key Passes
3.00
Activate Windows
Go to Settings to activate Windows.


VORN METRICS
Dashboard
Real-Time
Live Scores
Matchups
Fixtures
Match Predictions
League Table
Statistics
Teams
Players
Goalkeepers
Defenders
Midfielders
Forwards
Shots
2.50
Shooting Accuracy %
46.67%
Big Chances Missed
0.33
Aerials Won
0.33
Offsides
0.17
Big Chances Created
0.33
Crosses
6.67
Crossing Accuracy %
30.00%
Through Balls
0.17
Through Ball Accuracy %
0.00
Dribbling and Passing Stats
Dribbling Stats
Per Game
Dribble Attempts
3.33
Successful Dribbles
1.17
Dribble Success %
35.00%
Fouls Drawn
2.33
Passing Stats
Per Game
Passes
44.33
Accurate Passes
37.67
Accurate Passes %
84.96%
Long Balls
0.83
Activate Windows
Go to Settings to activate Windows.