# FANTASY PREMIER LEAGUE

# TEAM OPTIMIZATION USING

# GUIDED LEARNING MODELS AND SOCIAL MEDIA SENTIMENT ANALYSIS

*Review-2*

**B. Tech**
**Computer Science and Engineering**

**By**

**Rajarshi Saha (20BCT0163)**

**Ankan Ray (20BCE2304)**

*Under the guidance of*
*Dr.Ebenezer Juliet*
*Associate Professor Senior*
*SCOPE, VIT, Vellore*


**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**VELLORE INSTITUTE OF TECHNOLOGY**

**VELLORE 632 014, TAMILNADU, INDIA**

**April 2024**

# TABLE OF CONTENTS

## ABSTRACT

FPL (Fantasy Premier League) is one of the most popular fantasy leagues based on arguably the most popular sports annual event in the world i.e. the English Premier League. Sports Analytics has been a challenge for even the best of pundits and personal biasness and favoritism and also the human mind's inability to comprehend and account for the entire season and entirety of the 38 fixtures for each of the 20 teams in the league contribute to the poor performance of an individual and that's where data engineering, machine learning and artificial Intelligence can assist an individual in setting up the dream team. A Fantasy Premier league season however is not as straight-forward as there's a budget cap of 100 million for the FPL Managers, a max limit and minimum limit to players of each position and also from each team and a number of specific metrics that award players from different positions points. Moreover, there are Game weeks where a team has no fixtures and game weeks where a team has double fixtures and also there's a restriction of one free transfer per game week and 4 special game-chips: 1. Wildcard, 2. Bench Boost, 3. Free Hit, 4. Triple Captain. Hence, it is very important when building the team for game week 1 as it can make or break your season. Hence, we aim to build a platform that gives FPL managers insights by using visualizations, data engineering, ml and ai to provide predicted points for each player for each match, create best possible teams for the season and for the game week, transfer tips and build and train an AI model that can compete with other managers on FPL and make decisions.

# 1. INTRODUCTION

## 1.1 Introduction to the Project Domain

In the realm of Fantasy Premier League (FPL), the blend of data analytics and strategic decision-making is pivotal for success. Recognizing the challenges FPL managers face in player selection, fixture planning, and maximizing points within budget constraints, our platform aims to revolutionize team management.

By leveraging data engineering, machine learning, and artificial intelligence, our platform provides FPL managers with invaluable insights and predictive analytics. Our model meticulously plans for each game week, prioritizing fixtures based on team form, fixture difficulty ratings, and player fatigue.

We calculate predicted team scoring and conceding indices, enabling precise player point projections. For attackers and midfielders, factors such as team scoring indices, match fitness, and player form are considered. Defenders and goalkeepers are evaluated based on predicted team conceding indices and save performance.

Additionally, our platform incorporates a player popularity model, utilizing data from FPL Analytics and Twitter sentiment analysis to gauge player popularity and sentiment within the FPL community.

To optimize team selection, a statistical model predicts the best permutations of the playing XI and bench players, maximizing points while adhering to budget constraints. Time-series modelling ensures accurate predictions of player points over successive game weeks.

Through the integration of advanced analytics and intuitive interfaces, our platform aims to empower FPL managers, redefining the FPL experience and enabling managers to navigate the complexities of the Premier League with confidence and success.

## 1.2 Aim of the Project

The aim of the Fantasy Premier League (FPL) Assistant project is to revolutionize team management for FPL managers by leveraging data analytics, machine learning, and artificial intelligence. By addressing the challenges faced by FPL managers in player selection, fixture planning, and budget management, the aim is to redefine the FPL experience and empower managers to maximize their points throughout the season.

## 1.3 Objectives of the Project

- Develop a comprehensive platform that utilizes data engineering, machine learning, and artificial intelligence to analyze and predict player performance and points for each game week based on historical data and relevant factors.
- Assist FPL managers in making informed decisions regarding player selection, transfers, and team formation by providing actionable insights and recommendations tailored to individual team compositions and preferences.
- Prioritize fixtures and plan strategies for maximizing points while adhering to budget constraints, optimizing team selection and transfers for optimal performance.
- Enhance user experience and engagement by providing intuitive interfaces and personalized recommendations that improve usability and satisfaction.
- Continually improve and refine the platform through feedback and iteration, ensuring its relevance and effectiveness for FPL managers, thus contributing to ongoing success and satisfaction within the FPL community.

## 1.4 Scope of the Project

- Developing a comprehensive platform for Fantasy Premier League (FPL) managers.
- Incorporating data engineering, machine learning, and AI to provide insights and recommendations.
- Creating algorithms to predict player performance, optimize team selection, and suggest transfers.
- Integrating real-time data streams, including player statistics, fixture difficulty ratings, and betting market analysis.
- Implementing visualization tools to enhance user experience and facilitate decision-making.

- Building a user-friendly interface for FPL managers to access recommendations, transfer tips, and performance predictions.
- Conducting thorough testing and validation to ensure the accuracy and effectiveness of the platform.
- Continuously refining and updating the platform based on user feedback and performance analysis.
- Ultimately, empowering FPL managers to make informed decisions and maximize their success in the league.

## 1.5 Organization of the Thesis

# 2. LITERATURE REVIEW

## 2.1 Survey on Existing System

| | | |
|---|---|---|
| 1. **Player Recommendation System for Fantasy Premier League using Machine Learning** | **Vimal Rajesh, P Arjun, Kunal Ravikumar Jagtap, Suneera C M**<br><br>**2022**<br>**19th International Joint Conference on** | The paper aims to enhance decision-making for Fantasy Premier League (FPL) players by providing data-driven recommendations.<br><br>FPL is a popular online game where participants create virtual teams of real-life football players and earn points based on their performance in actual |

| | | |
|---|---|---|
| | **Computer Science and Software Engineering (JCSSE)** | matches.<br><br>The authors recognize the favouritism bias, where players tend to select players from their favourite teams rather than making informed choices.<br><br>They propose a recommendation system that leverages machine learning to suggest optimal player combinations.<br><br>The system uses data extracted from the FPL API.<br><br>The testing period corresponds to the English Premier League 2021–22 season.<br><br>The paper takes into consideration the Form, Return on Investment, Fixture Difficulty Rating, Bonus Points System, Points Per Game, and Influence Creativity Threat to determine the points of a player.<br><br>The paper proposes Random Forest and Gradient Boosting Machines to generate expected points for a player.<br><br>The paper uses Mean Absolute Error (MAE) is used as a metric to evaluate the model.<br><br>It explores statistical analysis and data science techniques to generate better recommendations. |
| 2. **Sports Analytics algorithms for performance prediction** | **Konstantinos Apostolou, Christos Tjortjis**<br><br>**2019,**<br>**10th International Conference on Information, Intelligence, Systems and Applications (IISA)** | The dissertation is divided into two major parts. The first part comprises a **literature review** of existing technologies related to sports analytics. The second part focuses on **experiments** conducted primarily using football data.<br><br>The first step involves gathering relevant data. In the context of football, this data could include player statistics, match results, team formations, and more.<br><br>Once collected, the data needs preprocessing. This step includes handling missing values, normalizing features, and ensuring data consistency.<br><br>Feature engineering is crucial for building effective predictive models. It involves creating new features or transforming existing ones to improve model performance.<br><br>For player performance prediction, features might |

| | | |
|---|---|---|
| | | include player position, playing time, goals scored, assists, passes completed, and defensive actions.<br><br>The model tries the following algorithms Naïve Bayes, Decision Tree, Random Forest, KNN, SVM (rbf kernel), SVM (poly kernel), XGBoost<br><br>The selected algorithm(s) are trained on the preprocessed data. The model learns patterns from historical data.<br><br>Evaluation metrics (e.g., accuracy, precision, recall) assess how well the model performs.<br><br>Once the model is trained, it can predict player performance based on input features.<br><br>Insights gained from predictions can guide team management decisions, such as player selection, substitutions, and tactical adjustments. |
| 3.<br><br>**Game Plan: What AI can do for Football, and What Football can do for AI** | **Karl Tuyls, Shayegan Omidshafiei, Paul Muller, Zhe Wang, Jerome Connor, Daniel Hennes**<br><br>**May 2021, Journal of Artificial Intelligence Research** | The paper is a theoretical discussion that sets the stage by emphasizing the unique synergy between artificial intelligence (AI) and the world's most popular sport, football.<br><br>It acknowledges that football analytics provides a rich playground for AI research due to its complexity, real-world dynamics, and massive data availability.<br><br>**Data-Driven Insights**: AI techniques enable data-driven insights into player performance, team strategies, and match outcomes.<br><br>**Predictive Models**: Algorithms predict player ratings, injury risks, and even transfer market values.<br><br>**Computer Vision**: AI-powered systems track player movements, ball trajectories, and tactical patterns during live matches.<br><br>**Fan Engagement**: Personalized content, interactive apps, and augmented reality experiences enhance fan engagement.<br><br>The paper discusses the following Challenges in Football Analytics: |

| | | |
|---|---|---|
| | | **High Dimensionality**: Football data is multidimensional, including player attributes, match events, and contextual factors.<br><br>**Temporal Dependencies**: Understanding how events unfold over time is crucial.<br><br>**Noise and Uncertainty**: Real-world data is noisy, incomplete, and subject to various uncertainties.<br><br>**Statistical Learning Techniques**:<br><br>**Regression Models**: Predicting continuous variables (e.g., player performance metrics).<br><br>**Classification Models**: Identifying player positions (e.g., forward, midfielder, defender).<br><br>**Clustering**: Grouping similar players based on playing style.<br><br>**Time Series Analysis**: Capturing temporal patterns in match data.<br><br>**Game Theory Applications**:<br><br>**Penalty Kicks**: Applying game theory to analyze optimal strategies for penalty takers and goalkeepers.<br><br>**Player Interactions**: Modeling interactions between players as strategic games.<br><br>**Computer Vision Challenges**:<br><br>**Player Tracking**: Extracting player trajectories from video feeds.<br><br>**Pose Estimation**: Inferring player body positions and orientations.<br><br>**Event Detection**: Recognizing goals, fouls, and other key events. |
| 4. | **Nicholas Bonello, Joeran Beel, Seamus** | **Fantasy Premier League (FPL)** performance predictors often rely solely on historical statistical |

| | | |
|---|---|---|
| **Multi-stream Data Analytics for Enhanced Performance Prediction in Fantasy Football** | **Lawless, Jeremy Debattista**<br><br>**2019, 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science** | data to predict player performances.<br><br>However, this approach overlooks external factors such as injuries, managerial decisions, and other tournament match statistics.<br><br>Traditional statistical predictors lack the ability to incorporate real-time information and human feedback.<br><br>Predictions based solely on historical data may not account for dynamic changes during a season.<br><br>The authors introduce a novel method that enhances performance prediction by **automatically incorporating human feedback** into the model.<br><br>They consider multiple streams of data beyond historical statistics to improve predictions.<br><br>**Data Streams Used**:<br><br>**Previous Performances**: Historical player performance data.<br><br>**Fixture Difficulty Ratings**: Upcoming fixture challenges.<br><br>**Betting Market Analysis**: Insights from betting odds.<br><br>**General Public and Expert Opinions**: Social media discussions, web articles, and expert insights.<br><br>The proposed model was tested on the **English Premier League 2018/19 season**.<br><br>It outperformed regular statistical predictors by over **300 points**, averaging **11 points per week**.<br><br>The model ranked within the top **0.5%** of players, achieving a rank of **30,000 out of over 6.5 million players**.<br><br>By incorporating diverse data streams and human feedback, this approach provides more robust and |

| | | |
|---|---|---|
| | | accurate predictions.<br><br>It demonstrates the potential of combining statistical analysis with real-world insights for fantasy football enthusiasts |
| 5.<br><br>**Football Player Value Assessment Using Machine Learning Techniques** | **Ahmet Talha Yiğit, Barış Samak and Tolga Kaya**<br><br>**2019, 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science** | **Sports analytics** is a growing field worldwide, and one of its open problems is the valuation of football players.<br><br>The aim of this study is to establish a **football player value assessment model** using machine learning techniques to support transfer decisions by football clubs.<br><br>The proposed models are mainly based on **intrinsic features** of individual players, as provided in the **Football Manager** video game.<br><br>The individual statistics of **5316 players** active in **11 different major leagues** from Europe and South America serve as the dataset.<br><br>The study employs advanced supervised learning techniques:<br><br>**Ridge and Lasso Regressions**: Regularized linear regression models.<br><br>**Random Forests**: Ensemble models combining decision trees.<br><br>**Extreme Gradient Boosting (XGBoost)**: Gradient boosting algorithm.<br><br>All models are built in the **R programming language**.<br><br>The models' performances are compared based on their **mean squared errors**.<br><br>An **ensemble model with inflation** is proposed as the output.<br><br>The goal is to value players based on their **normalized abilities**, relatively free from environmental variables. |

| | | |
|---|---|---|
| | | The model aims to provide better results than current models relying solely on in-field game statistics.<br><br>Considering the expansion of the football industry, a model using the latest developments in data analytics and machine learning addresses a significant problem for the industry.<br><br>It can be a valuable financial leverage for clubs seeking to expand their successes and profits.<br><br>This research contributes to the world of football by providing a data-driven approach to player valuation.<br><br>By leveraging machine learning techniques, clubs can make more informed transfer decisions. |
| 6.<br><br>**Fantasy Premier League - Performance Prediction** | **Pratik Pokharel, Arun Timalsina, Sanjeeb Panday, Bikram Acharya**<br><br>**2022, 12th IOE Graduate Conference** | This paper proposes a rational approach to player selection, team drafting, and transfers by predicting **ROI using xgboost regression**.<br><br>The study also evaluates the **impact of fixture congestion** on FPL points using mid-week cup fixture data.<br><br>Evaluation based on FPL global ranking reveals that initial drafted teams without transfers performed better than those with transfers, which were hindered by dependency on the accuracy of the regression model.<br><br>The **mean RMSE** score for all players was 2.048, and the effect of cup fixture congestion on FPL points was found to be insignificant.<br><br>The uncertainty of team selection makes it challenging for FPL managers, with Game-week 1 being crucial for laying the foundation of the season.<br><br>The paper highlights the vast number of potential team combinations and the difficulty in making optimal selections.<br><br>FPL analytics traditionally rely on historical statistical data but may overlook external factors such as **mid-week fixture fatigue** and **squad rotation strategies employed by managers** due to European and |

| | | |
|---|---|---|
| | | domestic cup competitions. |
| 7.<br><br>**Who Should Be the Captain This Week? Leveraging Inferred Diversity-Enhanced Crowd Wisdom for a Fantasy Premier League Captain Prediction** | **Shreyansh Bhatt, Keke Chen, Valerie L. Shalin, Amit P. Sheth, Brandon Minnery**<br><br>**2019, 13th International AAAI Conference on Web and Social Media (ICWSM)** | Utilizes the Wisdom of Crowd effect to predict productive players in Fantasy Sports.<br><br>Proposes the **SmartCrowd framework** to select a small, smart crowd using participants' Twitter posts.<br><br>Three main steps:<br><br>**characterizing participants using summary word vectors**<br><br>**clustering participants based on these vectors**<br><br>**sampling participants from clusters to form diverse crowds.**<br><br>Empirical evaluation on the Fantasy Premier League (FPL) captain prediction problem shows SmartCrowd outperforming random crowds and crowds consisting of top experts identified from previous performance data.<br><br>Social media-based diversity supports the sampling of smarter crowds that collectively predict productive players.<br><br>Studies the assembly of a small subset of the crowd using semantic diversity inferred from **participants' Twitter posts**.<br><br>Explores the weekly captain selection task in Fantasy Premier League (FPL) starting 11 player teams.<br><br>Utilizes crowd wisdom for determining parameters influencing captain choice.<br><br>Introduces the SmartCrowd approach, extending to other prediction problems, based on social media posts.<br><br>**Mines FPL user tweets** to infer crowd diversity based on topic and communication patterns.<br><br>Represents each Twitter user by a collection of their FPL tweets and summarizes them using **Word2vec**. |

| | | |
|---|---|---|
| | | **Tests multiple clustering strategies** and selects optimal representatives from clusters using **multi-objective optimization**.<br><br>Evaluates the approach using captain picks, points earned, and participants' previous seasons' performance scores.<br><br>Investigates questions related to the effectiveness of semantic analysis, diversity-based crowd selection, comparison with expert crowds, and the impact of diversity and crowd size. |
| 8.<br><br>**Time Series Modeling for Dream Team in Fantasy Premier League** | **Gupta, Akhil**<br><br>**2017, International Conference on Sports Engineering (ICSE)** | Utilizes data from the official Fantasy Premier League website for the past five to six years.<br><br>Cross-checks historical data using **Kaggle kernels** to ensure accuracy.<br><br>Employs data scraping techniques in Python 3 to collect two types of datasets: **Master FPL dataset and Points dataset** for each season.<br><br>Identifies and handles missing values in the datasets.<br><br>Scales down the cost field in the Master FPL dataset.<br><br>Creates a new ID field for matching datasets and merges points data into a single dataframe.<br><br>Removes players with incomplete historical data and those who left the league, ensuring data consistency.<br><br>Utilizes historical data to predict future player performance.<br><br>Implements two separate models**: ARIMA and LSTM-RNN**, which are later ensembled for improved accuracy.<br><br>Treats each player's performance data as a separate time series.<br><br>Validates models using data from previous seasons.<br><br>Formulates the problem of selecting the dream team as a **linear programming (LP)** problem. |

| | | |
|---|---|---|
| | | Seeks to maximize the total points of the selected players within budget constraints.<br><br>Utilizes the Pulp library in Python 2.7 for solving the LP problem.<br><br>Automates the prediction process after setting optimal parameters.<br><br>**Validates models using RMSE** and compares predicted values with actual values.<br><br>Determines optimal ensemble proportions for ARIMA and LSTM-RNN models.<br><br>Identifies the dream team for the upcoming season based on the optimized predictions.<br><br>Analyzes additional features such as **Points per Match (PPM), Cost per Point (CPP), and Cost-point index (CPI)** to evaluate player performance.<br><br>Discusses factors influencing player performance and team selection.<br><br>Highlights insights gained from the study, including player loyalty, nationality distribution, and team performance analysis.<br><br>Tracks the performance of the selected dream team for validation and further refinement of the approach. |
| 9.<br><br>**Football players performance analysis and formal/informal media: Sentiment analysis and semantic similarity** | **Gustavo Henrique de Sousa Silva, Rui Jorge Henriques Calado Lopes**<br><br>**2021, ISCTE** | The study utilized data from **informal media (e.g., Reddit)** and **formal media (e.g., Live Match commentary, Player Ratings)** to compare semantic similarity with key phrases from **Work Domain Analysis (WDA).**<br><br>Semantic Analysis involved comparing WDA key phrases with media entries using BERT to generate **vector representations** for textual data was used. Cosine similarity was computed between these representations.<br><br>Comments Processing: Each comment was subdivided |

| | | |
|---|---|---|
| | | into sentences.<br><br>Similarity Analysis: **BERT** was used to compare sentences with **WDA key phrases**, producing similarity scores ranging from 0 to 1.<br><br>Output Generation: **Heatmaps** were created to visualize the similarity values obtained.<br><br>**Linguistic Analysis**: Named entities and proper nouns were recognized to understand the main subjects of the match.<br><br>Sentiment Analysis: Polarity and subjectivity of comments were analyzed using TextBlob and Stanza libraries.<br><br>Output Generation: **Polarity** and **subjectivity** values were expressed in the range of -1 to 1 and 0 to 1, respectively.<br><br>The study employed Semantic Analysis to compare media entries with WDA key phrases and Sentiment Analysis to analyze the polarity and subjectivity of comments. These methodologies provided insights into the perception of football matches across different media sources. |
| 10.<br><br>**TF-Pundit: A Real-time Football Pundit based on Twitter** | **Karthik Anantha Padmanabhan** | **Actor Model Implementation**: The system, named **"TF-Pundit,"** is implemented using the Actor model, which facilitates concurrent computation. Actors receive messages and make decisions based on them, allowing the system components to execute concurrently.<br><br>**Text Processing Components**:<br><br>**Performance Analyzer**: Analyzes sentiment associated with players based on aggregated tweets. Utilizes a lexicon-based classifier to assign sentiment scores to players, considering words specific to football.<br><br>**Summarizer**: Summarizes football events based on tweets, discarding subjective opinions and |

| | | |
|---|---|---|
| | | selecting objective statements.<br><br>**Performance Analyzer Evaluation:** Compares ratings assigned by TF-Pundit with those from Goal.com, using a scaling formula. Scores are scaled between 0 and 5.<br><br>**Summarizer Evaluation:** Manually assigns scores to summaries on a scale of 1-10 based on how well they summarize one-minute intervals of football events.<br><br>The current implementation assumes that sentiment expressed in a tweet is solely for the mentioned player, which may not always be accurate. Tweets often express sentiments about multiple players simultaneously, leading to inaccuracies in sentiment analysis.<br><br>Generally, TF-Pundit's ratings were not significantly different from Goal.com ratings, indicating reasonable accuracy in assessing player performances.<br><br>Instances where discrepancies occurred were analyzed:<br><br>For example, a significant difference in the rating for "Shaarawy" was attributed to a specific event during the game that led to an inflated score due to numerous positive tweets.<br><br>This highlights the system's sensitivity to specific events and the need for context-aware analysis.<br><br>Prominent games, characterized by a higher volume of tweets and more noise, posed greater challenges for accurate player performance assessment. |

## 2.2 Gaps Identified

The literature review conducted revealed a gap in existing research pertaining to Fantasy Premier League (FPL) analysis. Specifically, none of the reviewed papers addressed the integration of past season performance analysis coupled with the consideration of team versus team history, which provides insight into opponent strength when assessing player performance. Furthermore, sentiment

analysis, a crucial aspect in understanding the contextual nuances surrounding player performance, was not explored in any of the identified literature. These findings underscore the opportunity to contribute novel insights and methodologies to the field of FPL analytics, particularly by incorporating historical performance data and sentiment analysis into predictive models for enhanced decision-making capabilities among FPL managers.

## 2.3Problem Statement

Creating a Fantasy Premier League (FPL) Assistant that optimizes team selection for FPL managers, considering insights from player performance in past sessions and already played matches, budget constraints, fixture prioritization, and sentiment analysis of players. This assistant aims to maximize the winning possibility of FPL managers by efficiently utilizing available resources and providing personalized recommendations tailored to individual team compositions and preferences.

# REQUIREMENT ANALYSIS

## 3.1 Requirements

### 3.1.1 Functional

1. Guided Learning Models:
    - The system should provide guided recommendations for team optimization based on machine learning models.
    - Machine learning algorithms should analyze historical player performance data and suggest optimal team configurations for upcoming matches.
2. **Social Media Sentiment Analysis**:
    - Integration with social media platforms to gather sentiment analysis data on FPL players.
    - Sentiment analysis should provide insights into the popularity and sentiment surrounding FPL players within the community.
3. **Fixture Prioritization**:
    - The system should prioritize fixtures based on team form, fixture difficulty ratings, and player fatigue.
    - Users should be provided with fixture analysis and recommendations to optimize team selection.
4. **Budget Management**:
    - Budget constraints should be enforced to ensure users adhere to FPL budget limits.
    - The system should suggest player transfers and substitutions to maximize team performance within budget constraints.
5. **Real-time Updates**:
    - Real-time updates on player performance, injuries, and other relevant news should

be provided to users.

- Users should be alerted to make informed decisions regarding team management.

### 3.1.2 Non-Functional

1. **Performance**:
    - The system should be responsive and provide fast performance, even during peak usage times.
    - Response times for recommendations and updates should be minimal to enhance user experience.
2. **Scalability**:
    - The system should be able to handle a large number of concurrent users without degradation in performance.
    - Scalability measures should be implemented to accommodate growth in user base and data volume.
3. **Security**:
    - User data, including personal information and login credentials, should be encrypted and stored securely.
    - Authentication mechanisms should be robust to prevent unauthorized access to user accounts.
4. **Reliability**:
    - The system should be highly reliable, with minimal downtime and errors.
    - Backup and recovery mechanisms should be in place to ensure data integrity and availability.
5. **Usability**:
    - The user interface should be intuitive and user-friendly, catering to users with varying levels of technical expertise.
    - Help and support features should be available to assist users in navigating the system.
6. **Privacy**:
    - User privacy should be protected, and data handling practices should comply with relevant privacy regulations.
    - Users should have control over their data and be able to manage privacy settings.
7. **Compatibility**:
    - The system should be compatible with a range of devices and web browsers to ensure accessibility for all users.
    - Compatibility with mobile devices should be prioritized to support users on-the-go.

## 3.2 Feasibility Study

### 3.2.1 Technical Feasibility

1. **Data Availability and Quality**: Assess the availability and quality of data required for implementing guided learning models and social media sentiment analysis. Ensure that sufficient and relevant data sources are accessible to train machine learning models and perform sentiment analysis effectively.
2. **Computational Resources**: Evaluate the computational resources needed to process large datasets and run machine learning algorithms efficiently. Consider factors such as server capacity, processing speed, and memory requirements to ensure the system can handle the computational demands of the project.
3. **Integration Capabilities**: Determine the feasibility of integrating with external APIs or platforms to access social media data and sentiment analysis tools. Ensure that APIs are well-documented, stable, and provide the necessary functionality for data retrieval and analysis.
4. **Scalability**: Assess the scalability of the system architecture to accommodate potential growth in user base and data volume over time. Design the system with scalability in mind, leveraging cloud-based solutions or scalable infrastructure components to handle increased load and demand.

### 3.2.2 Economic Feasibility

1. **Cost of Development**: Estimate the development costs associated with building the proposed system, including software development, data acquisition, and integration expenses. Consider the costs of hiring skilled personnel, acquiring necessary software licenses, and purchasing hardware infrastructure.
2. **Operational Costs**: Evaluate the ongoing operational costs of maintaining and running the system, including hosting fees, maintenance expenses, and any recurring subscription costs for external services or APIs. Ensure that the project remains financially viable and sustainable in the long term.
3. **Return on Investment (ROI)**: Assess the potential return on investment from the project, considering factors such as increased user engagement, subscription revenue, or advertising opportunities resulting from a successful implementation. Conduct a cost-benefit analysis to determine whether the expected benefits justify the investment.

### 3.2.3 Social Feasibility

1. User Acceptance: Evaluate the potential acceptance and adoption of the system by users, including FPL enthusiasts and fantasy sports fans. Conduct user surveys or focus groups to gather feedback on the proposed features and functionalities, ensuring that the system meets user expectations and preferences.
2. Community Engagement: Assess the potential impact of social media sentiment analysis on fostering community engagement and interaction among FPL managers.

Explore how sentiment analysis insights can enhance the user experience, facilitate discussions, and build a sense of community within the FPL ecosystem.

3. Ethical Considerations: Consider ethical implications related to data privacy, user consent, and responsible use of social media data in sentiment analysis. Ensure that the project adheres to ethical guidelines and respects user privacy rights, building trust and credibility among users and stakeholders.

## 3.3 System Specification

### 3.3.1 Hardware Specification

1. Memory (RAM): Sufficient RAM to accommodate data processing and machine learning model training tasks, depending on the size of the dataset and complexity of the models.
2. Storage: Adequate storage space to store datasets, trained models, and other project-related files. Consider using scalable storage solutions to accommodate potential growth in data volume.
3. Processing Power: Multi-core processors or high-performance CPUs to support parallelized data processing and model training tasks, reducing processing time and improving efficiency.

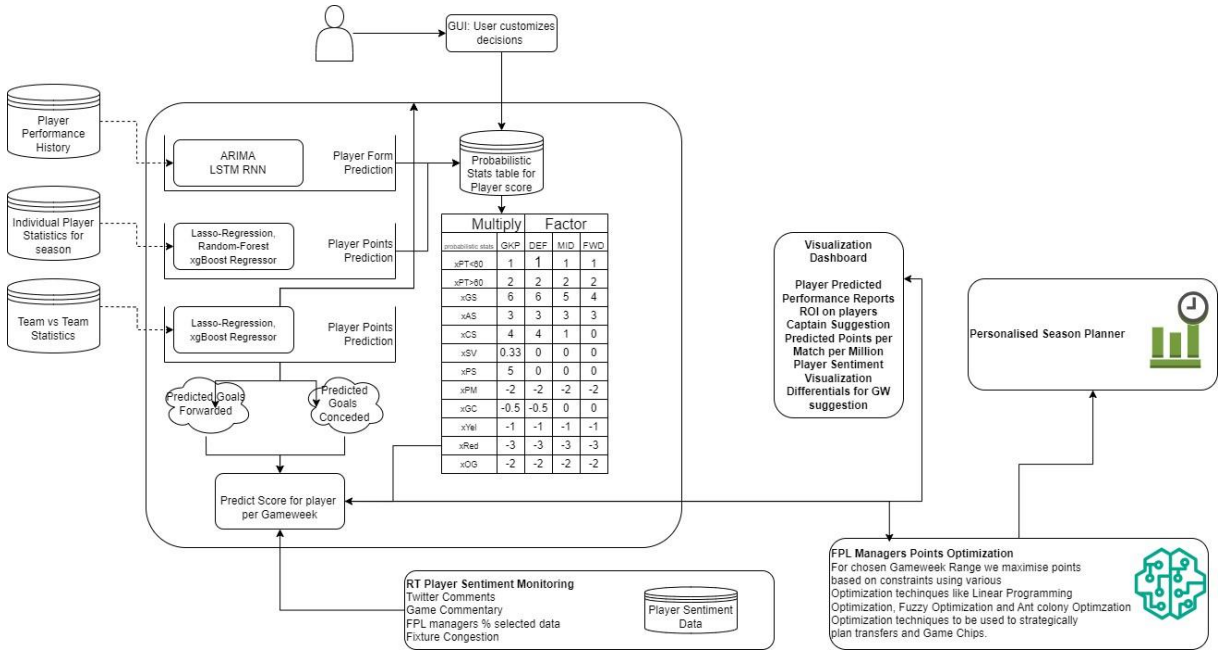
### 3.3.2 Software Specification

1. Programming Languages:
    - Python: For data analysis, machine learning model development, and web application development using frameworks like Flask or Django.
2. Data Analysis and Machine Learning Libraries:
    - Pandas: For data manipulation and analysis.
    - NumPy: For numerical computing and array operations.
    - Scikit-learn: For machine learning model development and evaluation.
    - Statsmodels: For time series analysis and statistical modeling.
3. Development Tools:
    - Jupyter Notebook or IDEs like PyCharm, VS Code, or Atom: For writing and executing Python code, data exploration, and model development.
    - Git: For version control and collaboration among team members.
4. External APIs and Libraries:
    - Twitter API or sentiment analysis libraries (e.g., NLTK, TextBlob, VADER): For accessing social media data and performing sentiment analysis, as mentioned in the project scope.

5. Operating System:
    - The software components listed above are compatible with various operating systems, including Windows, macOS, and Linux distributions, based on the team's preferences and requirements.

# 4. PROJECT DESCRIPTION

## 4.1 System Architecture



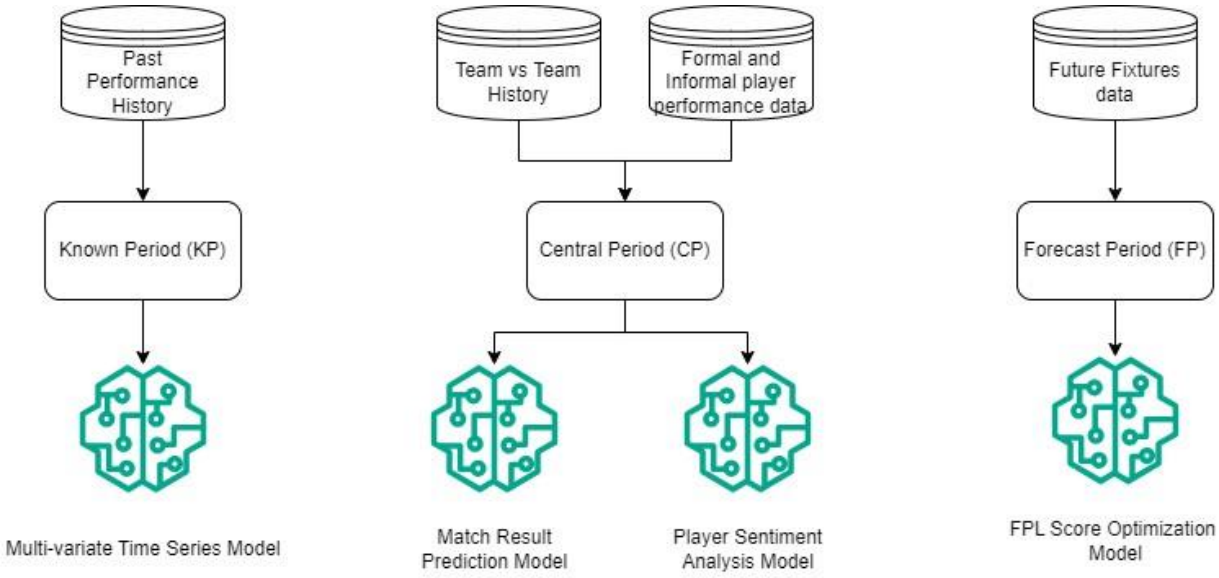

Before each Game week, the Model plans for future Game week fixtures with decreasing order of priority to each successive fixture, and for each player calculates Team Form and Predicted Team Scoring Index (xGfor) and Predicted Team Conceding Index (xGagnst) by using Form, Attack Index, Defence Index, Fixture Difficulty Rating FPL API data, previous history from FPLAnalytics.com and whoscored.com, time-series forecasting from previous season data and team fatigue Index from other Fixtures from whoscored.com.

Now, if the player is an Attacker/ Midfielder, we take the Predicted Team Score Index (xGfor) and the probable minutes per 90 index (xMP) and the match fitness (fatigue/ rest probability) and the Form of the player and the expected Scoring Index (xSc) and the expected Assisting Index (xAst) to calculate the probable points for the game (xPts).

If the player is Defender/ Goalkeeper, we take the inverse of Predicted Team Conceding Index (xGagnst) and the probable minutes per 90 index (xMP) and the match fitness (fatigue/ rest

probability) and the Form of the player and the expected save Index (xSv) and the discipline Index to calculate the probable points for the game.

For each player, we also consider a player Selection popularity Model where we using % selected, % captained data from FPLAnalystics and FPL API and using Twitter Sentiment Analysis on tweets on threads related to @FPLtweets for Sentiment analysis for players and formulate Popularity Metric using BERT (Bidirectional Encoder Representations from Transformers)



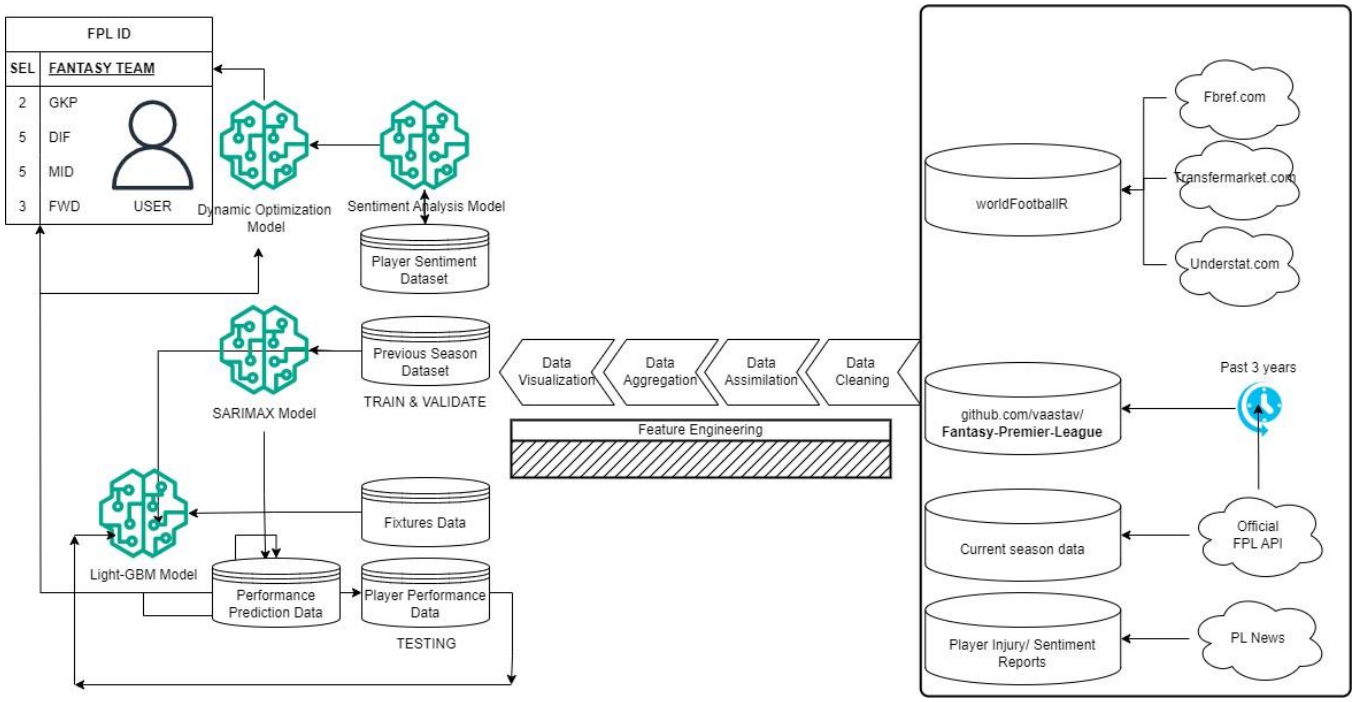

We then design a statistical model that predicts the best permutations of team (playing XI + 4 bench players) with formation using Predicted points per Cost Unit (Million) and create best possible team that maximizes the score for the upcoming fixtures.

For Time-Series Modelling we will be using a hybrid of Autoregressive Integrated Moving Average (ARIMA) known as SARIMAX to account for the seasonality for time series prediction and effect of exogenous regressors of player stats and subsequent maximization of total points using Linear Programming (LPP).

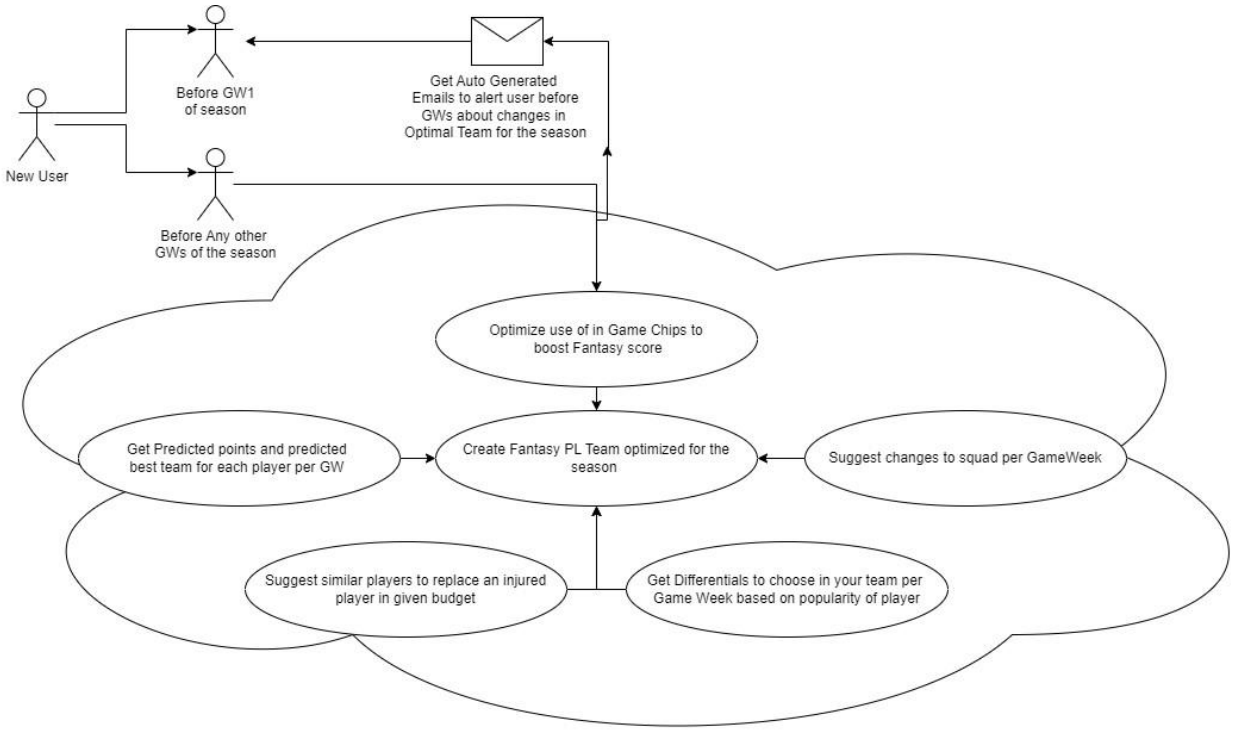
For Prediction of Fixture results and points for same season we will be using Random Forest Regressors with extreme Gradient Boosting (xgBoost). We have chosen LightGBM as an improvement over xgboost as LightGBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value while XGBoost uses pre-sorted algorithm & Histogram-based algorithm for computing the best split. We aim to feed trained data to an LLM model to create a chat interface for user with the data.

## 4.2 Design

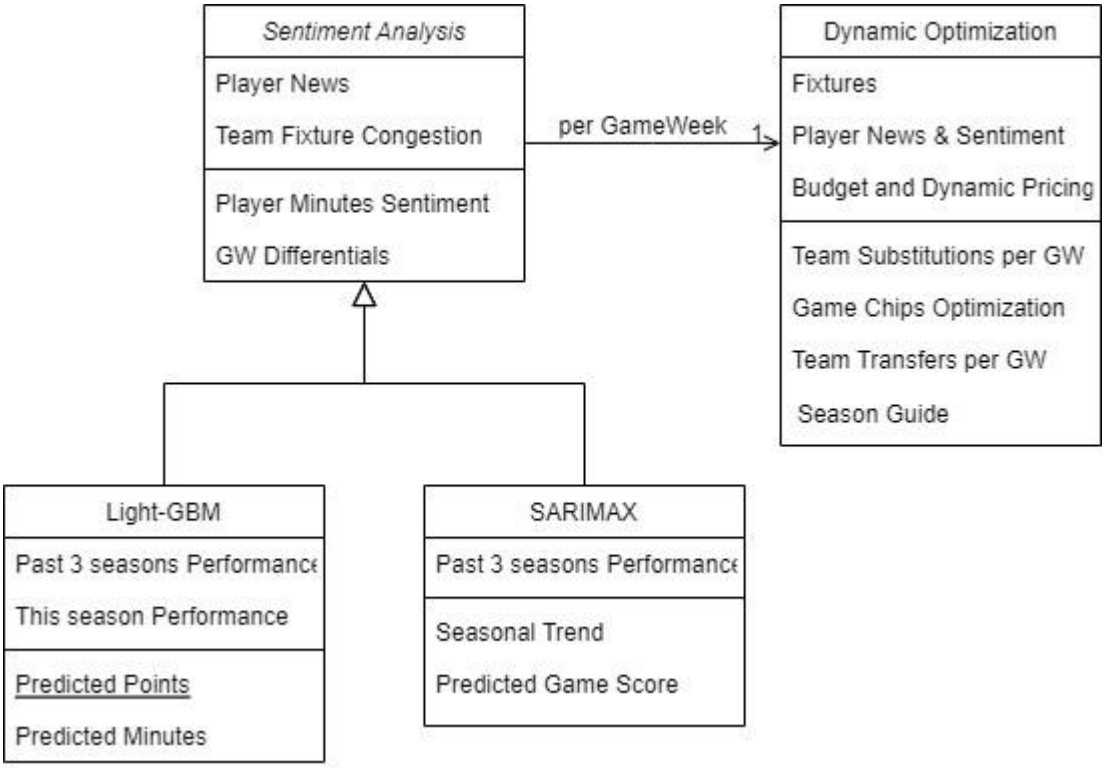
### 4.1.1 Data Flow Diagram



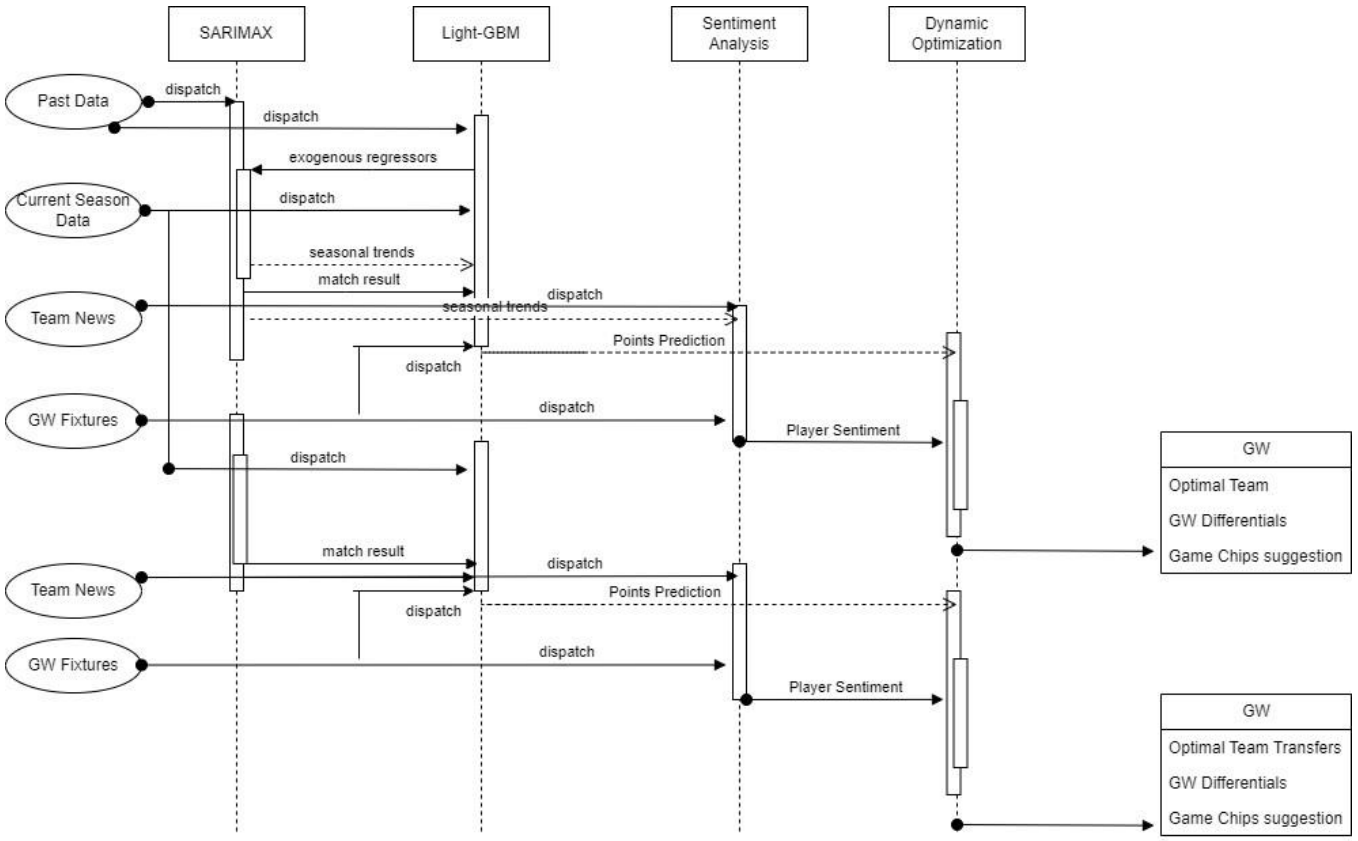

### 4.1.2 Use Case Diagram




### 4.1.3 Class Diagram




### 4.1.4 Sequence Diagram



## 4.3 Module Description

### 4.3.1 Data Scrapping:

- worldfootballR :

FBref.com - a whole host of data to analyse, including results, match stats, season long stats, player and team stats, etc.

Transfermarkt.com - player market values, team transfer history, player transfer history.

Understat.com - shot locations data for matches played in the major leagues.

- Fantasy Premier League API :

https://fantasy.premierleague.com/api

- Github Repository fir Premier League previous year statistics:

https://github.com/vaastav/Fantasy-Premier-League

### 4.3.2 Data Cleaning

- Removing Duplicates:

Involves identifying and eliminating duplicate records within the dataset to ensure data integrity and accuracy.

For our project, removing duplicates ensures that each player's performance statistics are uniquely represented, preventing any bias or distortion in the analysis and modeling process.

- Interpolating to Fill Missing Data:

Imputes missing values in the dataset by estimating them based on the surrounding data points.

For our project, interpolating missing data helps ensure completeness and consistency in player performance records, allowing for more comprehensive analysis and modeling.

- Mapping Teams to Team ID:

Establishes a relationship between team names and unique team identifiers (IDs)

for easy referencing and data consistency.

In our project, mapping teams to team IDs facilitates uniformity in representing team information across different datasets and sources, enhancing data integration and analysis capabilities.

- Mapping Players to Player ID:

  Associates player names with unique player identifiers (IDs) to standardize player references and enable seamless data linking and analysis.

  This mapping ensures that player performance data from various sources can be effectively integrated and analyzed within the project framework.

- Merging Season Stats:

  Combines player performance statistics from different seasons into a unified dataset, allowing for comprehensive analysis and trend identification across multiple seasons.

Merging season stats involves aggregating player performance metrics such as goals scored, assists, clean sheets, etc., from individual seasons into a single dataset. This aggregated dataset provides a holistic view of player performance over time, enabling the development of more robust models and strategies for FPL team optimization. Additionally, by merging season stats, we can identify patterns, trends, and player consistency across multiple seasons, offering valuable insights for FPL managers in team selection and strategy planning

### 4.3.3 Data Assimilation

1. Merging Opponent Data:
   - Involves combining information about opponent teams with player performance data to enrich the dataset.
   - This merging enables the analysis of player performance in the context of the opponents they faced, providing insights into player performance variations based on the strength of the opposition.
2. Creating Mapping Between Tables:
   - Establishes relationships between different tables or datasets by defining key identifiers that link corresponding records.
   - This mapping facilitates data integration and retrieval, allowing for seamless analysis and querying across multiple tables within the database.
3. Converting from Raw API Data to Structured Data:
   - Refers to the process of transforming raw data obtained from APIs into a structured format suitable for analysis and storage.
   - This conversion involves parsing, cleaning, and organizing the raw data

into tables or datasets with well-defined schemas, making it easier to analyze and extract insights.

4. Combining Data Collected from Various Sources:
    - Integrates data obtained from diverse sources, such as APIs, databases, files, or web scraping, into a unified dataset.
    - By combining data from multiple sources, we can leverage complementary information to enrich analysis, gain a comprehensive understanding of the subject matter, and uncover hidden patterns or correlations.

### 4.3.4 Feature Engineering

- Last N Week Stats Calculation:

  Implemented in functions get_last_stats, get_players_last_stats_test, and get_all_players_last_stats.Calculates statistics for each player from the last N gameweeks.The statistics considered are specified in the history_stats list.

- Mean and Standard Deviation Calculation:
    - Implemented in function create_features.
    - Calculates the mean and standard deviation of selected statistics (mean_features and std_features) from the last N gameweeks.
    - Appends these mean and standard deviation features to the dataframe.

- Percentage Value to Team and Position Rank Calculation:
    - Implemented within the loop iterating over each gameweek.
    - Calculates the percentage value of each player relative to the total value of their team.
    - Calculates the position rank of each player within their position and team based on value.
- Opponent Team and Last Season Position Extraction:
    - Implemented in the loop iterating over each year and gameweek.
    - Extracts the opponent team for each match and their position in the previous season.
    - Merges this information with the main dataframe.

### 4.3.5 Model Training

1. SARIMAX - Seasonal AutoRegressive Integrated Moving Average with eXogenous factors is a time series forecasting model that extends the traditional ARIMA model to handle seasonality and exogenous variables.

$$d_t = c + \sum_{n=1}^{p} \alpha_n d_{t-n} + \sum_{n=1}^{q} \theta_n \epsilon_{t-n} + \sum_{n=1}^{r} \beta_n x_{n_t} + \sum_{n=1}^{P} \phi_n d_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t$$

- Discrete Fourier Transform testing has been conducted. DFT testing is a method used to analyse periodic patterns or seasonality in time series data. By completing DFT testing, the model has assessed the presence and characteristics of seasonality in the data.
- Model data assimilated implying that the model has incorporated and integrated relevant data. This likely includes historical time series data, exogenous variables (if applicable), and any other relevant information used in model training and analysis.

2. Light GBM - Light Gradient Boosting Machine is a powerful and efficient gradient boosting framework used for supervised learning tasks such as classification, regression, and ranking. It is designed to be highly scalable and can handle large datasets with millions of instances and features. Gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages: Faster training speed and higher efficiency, Lower memory usage, better accuracy, Support of parallel, distributed, and GPU learning and Capable of handling large-scale data.
    - Trained on Last 3 seasons
    - Trained on First 10 Game weeks for the new season

# 5. SAMPLE CODE

```
import numpy as np
import pandas as pd
from scipy import stats


def split_test(data, gameweek):
    # print(data["GW"].value_counts())
    data_gw = data[data["GW"] == gameweek]
    data_other_gw = data[~(data["GW"] == gameweek)]
    return data_gw, data_other_gw


def check_win(df):
    list_win = []
    for index in df.index:
        result = (df["team_a_score"] - df["team_h_score"]).loc[index]
        is_home = df["was_home"].loc[index]
        if result == 0:
            list_win.append(1)
```

```
        elif result > 0 and is_home == True:
            list_win.append(0)
        elif result < 0 and is_home == True:
            list_win.append(3)
        elif result > 0 and is_home == False:
            list_win.append(3)
        elif result < 0 and is_home == False:
            list_win.append(0)
        else:
            list_win.append(-1)
    return list_win


def get_2020_21_season_pos(club):
    """get the position of the club in the 2020-21 season"""
    if club == "Man City":
        return 1
    elif club == "Man Utd":
        return 2
    elif club == "Liverpool":
        return 3
    elif club == "Chelsea":
        return 4
    elif club == "Leicester":
        return 5
    elif club == "West Ham":
        return 6
    elif club == "Spurs":
        return 7
    elif club == "Arsenal":
        return 8
    elif club == "Leeds":
        return 9
    elif club == "Everton":
        return 10
    elif club == "Aston Villa":
        return 11
    elif club == "Newcastle":
        return 12
    elif club == "Wolves":
        return 13
    elif club == "Crystal Palace":
        return 14
    elif club == "Southampton":
        return 15
    elif club == "Brighton":
        return 16
    elif club == "Burnley":
        return 17
    else:
        return 20
```

```
def get_2019_20_season_pos(club):
    """get the position of the club in the 2019-20 season"""
    if club == "Liverpool":
        return 1
    elif club == "Man City":
        return 2
    elif club == "Man Utd":
        return 3
    elif club == "Chelsea":
        return 4
    elif club == "Leicester":
        return 5
    elif club == "Spurs":
        return 6
    elif club == "Wolves":
        return 7
    elif club == "Arsenal":
        return 8
    elif club == "Sheffield Utd":
        return 9
    elif club == "Burnley":
        return 10
    elif club == "Southampton":
        return 11
    elif club == "Everton":
        return 12
    elif club == "Newcastle":
        return 13
    elif club == "Crystal Palace":
        return 14
    elif club == "Brighton":
        return 15
    elif club == "West Ham":
        return 16
    elif club == "Aston Villa":
        return 17
    else:
        return 20


def get_2021_22_season_pos(club):
    if club == "Man City":
        return 1
    elif club == "Liverpool":
        return 2
    elif club == "Chelsea":
        return 3
    elif club == "Spurs":
        return 4
    elif club == "Arsenal":
        return 5
```

```
    elif club == "Man Utd":
        return 6
    elif club == "West Ham":
        return 7
    elif club == "Leicester":
        return 8
    elif club == "Brighton":
        return 9
    elif club == "Wolves":
        return 10
    elif club == "Newcastle":
        return 11
    elif club == "Crystal Palace":
        return 12
    elif club == "Brentford":
        return 13
    elif club == "Aston Villa":
        return 14
    elif club == "Southampton":
        return 15
    elif club == "Everton":
        return 16
    elif club == "Leeds":
        return 17
    else:
        return 20


def get_2022_23_season_pos(club):
    if club == "Man City":
        return 1
    elif club == "Arsenal":
        return 2
    elif club == "Man Utd":
        return 3
    elif club == "Newcastle":
        return 4
    elif club == "Liverpool":
        return 5
    elif club == "Brighton":
        return 6
    elif club == "Aston Villa":
        return 7
    elif club == "Spurs":
        return 8
    elif club == "Brentford":
        return 9
    elif club == "Fulham":
        return 10
    elif club == "Crystal Palace":
        return 11
    elif club == "Chelsea":
```

```
        return 12
    elif club == "Wolves":
        return 13
    elif club == "West Ham":
        return 14
    elif club == "Bournemouth":
        return 15
    elif club == "Nott'm Forest":
        return 16
    elif club == "Everton":
        return 17
    else:
        return 20


def get_last_season_pos(year):
    """get the function to get the last season position of a team at any year"""
    if year == "2020-21":
        return get_2019_20_season_pos
    elif year == "2021-22":
        return get_2020_21_season_pos
    elif year == "2022-23":
        return get_2021_22_season_pos
    elif year == "2023-24":
        return get_2022_23_season_pos


def remove_neg(val):
    if val > 0:
        return val
    else:
        return -val


def deque_and_queue(stats, value):
    # if -1 in stats:
    #  return stats
    # deque
    stats = stats[1:]
    stats.append(value)
    return stats

def get_all_stats(data, stat, name):
    '''All stats for a player'''
    name_df = data[data["name"] == name]
    seasons = ["2020-21", "2021-22", "2022-23"]
    name_df_dict = {}
    for season in seasons:
        name_df_season = name_df[name_df["season"] == season]
        list_stats = []
        stats_x = []
        for value in name_df_season[stat]:
```

```
            list_stats.append(np.array(stats_x))
            stats_x.append(value)


        name_df_season[f"all {stat}"] = list_stats
        name_df_dict[season] = name_df_season
    return pd.concat(name_df_dict.values())


def get_last_stats(data, stat, name, no_last_stats=short_term_stats):
    '''Get last n week stats for specific player'''
    name_df = data[data["name"] == name]
    seasons = ["2020-21", "2021-22", "2022-23"]
    name_df_dict = {}
    for season in seasons:
        name_df_season = name_df[name_df["season"] == season]
        list_stats = []
        stats_x = []
        for value in name_df_season[stat]:
            if len(stats_x) < no_last_stats:
                list_stats.append(np.array(stats_x))
                stats_x.append(value)
            else:
                list_stats.append(np.array(stats_x))
                stats_x = deque_and_queue(stats_x, value)
        name_df_season[f"last {no_last_stats} {stat}"] = list_stats
        name_df_dict[season] = name_df_season
    return pd.concat(name_df_dict.values())


def get_all_players_last_stats(data, stat, no_last_stats=short_term_stats):
    '''Get last n week stats for all players'''
    players_df = []
    for player in data["name"].unique():
        data_player = get_last_stats(data, stat, player, no_last_stats)
        players_df.append(data_player)
    # print(pd.concat(players_df))
    return pd.concat(players_df)

def get_all_players_all_stats(data, stat):
    '''Get all stats for all players'''
    players_df = []
    for player in data["name"].unique():
        data_player =  get_all_stats(data, stat, player)
        players_df.append(data_player)
    # print(pd.concat(players_df))
    return pd.concat(players_df)

def get_last_stats_test(data, stat, name, no_last_stats=short_term_stats):
    '''Get last n week stats for specific players'''
    name_df = data[data["name"] == name]
    list_stats = []
```

```
    stats_x = []
    for value in name_df[stat]:
        if len(stats_x) < no_last_stats:
            list_stats.append(np.array(stats_x))
            stats_x.append(value)
        else:
            list_stats.append(np.array(stats_x))
            stats_x = deque_and_queue(stats_x, value)
    name_df[f"last {no_last_stats} {stat}"] = list_stats
    return name_df


def get_players_last_stats_test(data, stat, no_last_stats=short_term_stats):
    '''Get last n week stats for all players'''
    players_df = []
    for player in data["name"].unique():
        data_player = get_last_stats_test(data, stat, player, no_last_stats)
        players_df.append(data_player)
    # print(pd.concat(players_df))
    return pd.concat(players_df)

def get_all_stats_test(data, stat, name):
    '''Get all stats for specific player'''
    name_df = data[data["name"] == name]
    list_stats = []
    stats_x = []
    for value in name_df[stat]:
        list_stats.append(np.array(stats_x))
        stats_x.append(value)
    name_df[f"all {stat}"] = list_stats
    return name_df


def get_players_all_stats_test(data, stat):
    '''Get all stats for all player'''
    players_df = []
    for player in data["name"].unique():
        data_player = get_all_stats_test(data, stat, player)
        players_df.append(data_player)
    # print(pd.concat(players_df))
    return pd.concat(players_df)


def convert_minutes(val):
    """CONVERTS MINUTES TO A CATEGORICAL OUTPUT"""
    if val > 10:
        return 1
    else:
        return 0


def find_mode(vals):
```

```
    """find the mode of vals"""
    try:
        if -1 in vals:
            return -1
        return stats.mode(vals)[0][0]
    except IndexError:
        return np.nan


def find_mean(vals):
    """find the mean of vals"""
    try:
        if -1 in vals:
            return -1
        return np.mean(vals)
    except:
        return np.nan


def find_max(vals):
    """find the maximum of vals"""
    try:
        if -1 in vals:
            return -1
        return np.max(vals)
    except:
        return np.nan


def find_std(vals):
    """find the standard deviation of vals"""
    try:
        if -1 in vals:
            return -1
        return np.std(vals)
    except:
        return np.nan


def find_value_count(vals, to_count):
    """find the number of times to_count appears in vals"""
    try:
        if -1 in vals:
            return -1
        values, count = np.unique(vals, return_counts=True)
        index = np.where(values == to_count)[0][0]
        return count[index]

    except:
        return -2
```

```
def get_opp_team(data):
    """get the opponent team for each player"""
    opp_teams = []
    home_teams = data["home_team"]
    away_teams = data["away_team"]
    my_teams = data["team"]
    for id in data.index:
        my_team = my_teams.iloc[id]
        home_team = home_teams.iloc[id]
        away_team = away_teams.iloc[id]
        if my_team == home_team:
            opp_teams.append(away_team)
        else:
            opp_teams.append(home_team)
    return opp_teams



def create_features(data, mean_features, std_features):
    for mean_feature in mean_features:
        print(mean_feature)
        data[f"mean {mean_feature} {no_last_stats}"] = [
            find_mean(values) for values in data[f"last {no_last_stats}
{mean_feature}"]
        ]
        data[f"mean {mean_feature} all"] = [
            find_mean(values) for values in data[f"all {mean_feature}"]
        ]
        #data[f"mean {mean_feature} all"]= data[f"mean {mean_feature}
all"]/data["GW"]
    for std_feature in std_features:
        print(std_feature)
        data[f"std {std_feature} {no_last_stats}"] = [
            find_std(values) for values in data[f"last {no_last_stats}
{std_feature}"]
        ]
    return data


#from config import unused_columns, history_stats, no_last_stats
import warnings

warnings.filterwarnings("ignore")


def calculate_ratio_team_value(name, df):
    """Calculate the ratio of player value to team value"""
    team = df[df["name"] == name]["team"].iloc[0]
    total_value = df[df["team"] == team]["value"].sum()
    value = df[df["name"] == name]["value"].iloc[0]
    return value * 100 / total_value
```

```
def calculate_position_rank(name, df):
    """Calculate the number of players with a higher value"""
    value = df[df["name"] == name]["value"].iloc[0]
    position = df[df["name"] == name]["position"].iloc[0]
    team = df[df["name"] == name]["team"].iloc[0]
    return df[
        (df["value"] > value) & (df["position"] == position) & (df["team"] == team)
    ]["value"].shape[0]



list_dfs = []

for i, year in enumerate(years):
    print(year)

    # get previous_seasons_data
    player_prev_stats = pd.read_csv(
        f"https://raw.githubusercontent.com/vaastav/Fantasy-Premier-
League/master/data/{previous_years[i]}/cleaned_players.csv"
    )
    player_prev_stats["name"] = (
        player_prev_stats["first_name"] + " " + player_prev_stats["second_name"]
    )
    player_prev_stats.drop(["first_name", "second_name"], axis=1, inplace=True)
    player_prev_stats.columns = player_prev_stats.columns + "_ex"

    # get opponent_team
    teams = pd.read_csv(
        f"https://raw.githubusercontent.com/vaastav/Fantasy-Premier-
League/master/data/{year}/teams.csv",
        encoding="latin-1",
    )[["id", "name"]]
    teams.columns = ["opponent_team", "opponent"]

    # opponents position last season
    teams["opponent_last_season_position"] = teams["opponent"].apply(
        get_last_season_pos(year)
    )
    for gameweek in range(1, 39):
        print(gameweek)
        df = pd.read_csv(
            f"https://raw.githubusercontent.com/vaastav/Fantasy-Premier-
League/master/data/{year}/gws/gw{gameweek}.csv",
            encoding="latin-1",
        )

        # teams position last season
        df["last_season_position"] = df["team"].apply(get_last_season_pos(year))
```

```
        # calculate percentage value to team
        pv_list = []
        pr_list = []
        for name in df["name"]:
            pv_list.append(calculate_ratio_team_value(name, df))
            pr_list.append(calculate_position_rank(name, df))
        df["percent_value"] = pv_list
        df["position rank"] = pr_list

        # chek if the result was a win or not
        df["match_result"] = check_win(df)

        # merge previous_season_data
        df = pd.merge(
            df, player_prev_stats, left_on="name", right_on="name_ex", how="left"
        )
        df["season"] = year
        df.drop("name_ex", axis=1, inplace=True)
        df["GW"] = gameweek

        # merge opponent team
        df = pd.merge(df, teams, on="opponent_team", how="left")
        list_dfs.append(df)


all_data = pd.concat(list_dfs)
# all_data.to_csv("/kaggle/working/previous_seasons.csv")
```

# 6. RESULTS OBSERVED

```
test[test["position"]=="MID"].sort_values(by="points",ascending=False).head(11)[["name","opponent_last_se
```

[188...

| index | name | opponent_last_season_position | was_home | points | team | value |
|---|---|---|---|---|---|---|
| Mohamed Salah2023-08-13T15:30:00Z | Mohamed Salah | 12 | 1 | 10.182815 | Liverpool | 125 |
| Martin Ødegaard2023-08-12T12:00:00Z | Martin Ødegaard | 16 | 0 | 6.500668 | Arsenal | 85 |
| Bukayo Saka2023-08-12T12:00:00Z | Bukayo Saka | 16 | 0 | 6.240895 | Arsenal | 85 |
| Bruno Borges Fernandes2023-08-14T19:00:00Z | Bruno Borges Fernandes | 13 | 0 | 5.791867 | Man Utd | 85 |
| Kevin De Bruyne2023-08-11T19:00:00Z | Kevin De Bruyne | 20 | 1 | 5.638430 | Man City | 105 |
| Gabriel Martinelli Silva2023-08-12T12:00:00Z | Gabriel Martinelli Silva | 16 | 0 | 5.130547 | Arsenal | 80 |
| Marcus Rashford2023-08-14T19:00:00Z | Marcus Rashford | 13 | 0 | 4.194368 | Man Utd | 90 |
| Pascal Groß2023-08-12T14:00:00Z | Pascal Groß | 20 | 0 | 4.002104 | Brighton | 65 |
| Solly March2023-08-12T14:00:00Z | Solly March | 20 | 0 | 3.555792 | Brighton | 65 |
| Eberechi Eze2023-08-12T14:00:00Z | Eberechi Eze | 20 | 1 | 3.088195 | Crystal Palace | 65 |
| Son Heung-min2023-08-13T13:00:00Z | Son Heung-min | 9 | 1 | 3.034857 | Spurs | 90 |

[191]:
```
test[test["position"]=="GKP"].sort_values(by="points",ascending=False).head(10)[["name","opponent_last_s
```

[191…

| index | name | opponent_last_season_position | was_home | points | team | value |
|---|---|---|---|---|---|---|
| Aaron Ramsdale2023-08-12T12:00:00Z | Aaron Ramsdale | 16 | 0 | 3.653835 | Arsenal | 50 |
| Alisson Ramses Becker2023-08-13T15:30:00Z | Alisson Ramses Becker | 12 | 1 | 3.400355 | Liverpool | 55 |
| David Raya Martin2023-08-13T13:00:00Z | David Raya Martin | 9 | 0 | 3.256289 | Arsenal | 50 |
| José Malheiro de Sá2023-08-14T19:00:00Z | José Malheiro de Sá | 3 | 1 | 3.143887 | Wolves | 50 |
| Emiliano Martínez Romero2023-08-12T16:30:00Z | Emiliano Martínez Romero | 4 | 1 | 2.944158 | Aston Villa | 50 |
| Nick Pope2023-08-12T16:30:00Z | Nick Pope | 7 | 0 | 2.715660 | Newcastle | 55 |
| Lukasz Fabianski2023-08-12T14:00:00Z | Lukasz Fabianski | 15 | 1 | 2.629725 | West Ham | 45 |
| Ederson Santana de Moraes2023-08-11T19:00:00Z | Ederson Santana de Moraes | 20 | 1 | 2.539753 | Man City | 55 |
| Bernd Leno2023-08-12T14:00:00Z | Bernd Leno | 17 | 1 | 2.145955 | Fulham | 45 |
| Jordan Pickford2023-08-12T14:00:00Z | Jordan Pickford | 10 | 0 | 2.092276 | Everton | 45 |

```
test[test["position"]=="DEF"].sort_values(by="points",ascending=False).head(10)[["name","opponent_last_seas
```

[189…

| index | name | opponent_last_season_position | was_home | points | team | value |
|---|---|---|---|---|---|---|
| Kieran Trippier2023-08-12T16:30:00Z | Kieran Trippier | 7 | 0 | 4.389531 | Newcastle | 65 |
| Trent Alexander-Arnold2023-08-13T15:30:00Z | Trent Alexander-Arnold | 12 | 1 | 4.257807 | Liverpool | 80 |
| Andrew Robertson2023-08-13T15:30:00Z | Andrew Robertson | 12 | 1 | 2.742380 | Liverpool | 65 |
| Ben Mee2023-08-13T13:00:00Z | Ben Mee | 8 | 0 | 2.606454 | Brentford | 50 |
| Pervis Estupiñán2023-08-12T14:00:00Z | Pervis Estupiñán | 20 | 0 | 2.451813 | Brighton | 50 |
| Virgil van Dijk2023-08-13T15:30:00Z | Virgil van Dijk | 12 | 1 | 2.421160 | Liverpool | 60 |
| Lewis Dunk2023-08-12T14:00:00Z | Lewis Dunk | 20 | 0 | 2.408961 | Brighton | 50 |
| Lisandro Martínez2023-08-14T19:00:00Z | Lisandro Martínez | 13 | 0 | 2.341776 | Man Utd | 50 |
| William Saliba2023-08-12T12:00:00Z | William Saliba | 16 | 0 | 2.321176 | Arsenal | 50 |
| Gabriel dos Santos Magalhães2023-08-12T12:00:00Z | Gabriel dos Santos Magalhães | 16 | 0 | 2.318982 | Arsenal | 50 |

[192]:
```
test[test["position"]=="FWD"].sort_values(by="points",ascending=False).head(10)[["name","opponent_last_sea
```

[192…

| index | name | opponent_last_season_position | was_home | points | team | value |
|---|---|---|---|---|---|---|
| Erling Haaland2023-08-11T19:00:00Z | Erling Haaland | 20 | 1 | 10.194150 | Man City | 140 |
| Harry Kane2023-08-13T13:00:00Z | Harry Kane | 9 | 1 | 9.730697 | Spurs | 125 |
| Ivan Toney2023-08-13T13:00:00Z | Ivan Toney | 8 | 0 | 4.462744 | Brentford | 80 |
| Aleksandar Mitrović2023-08-12T14:00:00Z | Aleksandar Mitrović | 17 | 1 | 3.465743 | Fulham | 75 |
| Gabriel Fernando de Jesus2023-08-12T12:00:00Z | Gabriel Fernando de Jesus | 16 | 0 | 3.355168 | Arsenal | 80 |
| Dominic Solanke2023-08-12T14:00:00Z | Dominic Solanke | 14 | 0 | 3.088639 | Bournemouth | 65 |
| Ollie Watkins2023-08-12T16:30:00Z | Ollie Watkins | 4 | 1 | 2.984169 | Aston Villa | 80 |
| Callum Wilson2023-08-12T16:30:00Z | Callum Wilson | 7 | 0 | 2.849204 | Newcastle | 80 |
| Darwin Núñez Ribeiro2023-08-13T15:30:00Z | Darwin Núñez Ribeiro | 12 | 1 | 2.730184 | Liverpool | 75 |
| Alexander Isak2023-08-12T16:30:00Z | Alexander Isak | 7 | 0 | 2.350178 | Newcastle | 75 |