# Fantasy Football Prediction System: A Comprehensive Report

Deban Kumar Shahi, Piyush Priy, Sanket Adlak, Priyank Nagarnaik

May 8, 2025

## 1 Introduction

Fantasy Premier League (FPL) is a global phenomenon, engaging millions of players who aim to build a competitive 15-player squad and an 11-player starting lineup within a £100m budget, adhering to position constraints (2 goalkeepers, 5 defenders, 5 midfielders, 3 forwards) and a maximum of three players per Premier League team[1]. The challenge lies in predicting player performance, which is influenced by goals, assists, clean sheets, and other metrics, while avoiding human biases such as over-reliance on popular players or recent high scores[1]. For instance, a player like Mohamed Salah might be selected based on reputation, even if facing a tough opponent, leading to suboptimal points[1].

This project addresses these challenges by developing a Fantasy Football Prediction System that leverages machine learning and optimization techniques to predict player performance and recommend optimal lineups. The system achieved a top 16.6% ranking, scoring 1364 points over 25 gameweeks compared to the average of 1262 points, demonstrating its effectiveness[1]. The methodology involves data collection, preprocessing, feature engineering, data filtering, model training with regression algorithms (Linear, Ridge, Lasso, ElasticNet), and team selection optimization using PuLP's Binary Integer Linear Programming (BILP) framework[1].

This report provides a comprehensive overview of the system, detailing each component of the pipeline. Intended visualizations, such as a data filtering flowchart, model performance comparisons, feature importance plots, and team selection confusion matrices, are described as placeholders to ensure compilation without external dependencies. The report also covers experimental validation, results, and implications for broader applications, with a focus on interpretability and scalability[1].

## 2 Methodology

The methodology encompasses several stages: data collection, preprocessing, filtering, feature engineering, model training, and team selection optimization. Each step is designed to maximize predictive accuracy and optimize squad selection under FPL constraints.

[1]github: https://github.com/SanketAdlak/FPLpredictor.

## 2.1 Data Collection and Preprocessing

Data was sourced from the Vaastav FPL repository, covering seasons 2019/20 to 2024/25, including merged gameweek files and player-level data[1]. The training dataset spans 2021–22 to 2023–24, with the 2024–25 season reserved for testing. The raw dataset comprises 103,801 rows (player-gameweek combinations) and 46 columns, including metrics like `total_points`, `goals_scored`, `assists`, `minutes`, and `opponent_team`[1].

Preprocessing involved several steps to ensure data consistency: - **Player Name Standardization**: Player names were cleaned to remove inconsistencies (e.g., "Jamie_Vardy_166" to "Jamie Vardy") to enable accurate grouping across gameweeks[1]. - **Position Mapping**: The `element_type` field was mapped to positions (e.g., 1 to GK, 2 to DEF, 3 to MID, 4 to FWD) to align with FPL's position categories[1]. - **Missing Value Imputation**: For features like `ict_index`, missing values were imputed with the mean value for that position to avoid bias in downstream modeling[1]. - **Categorical Encoding**: Binary features like `was_home` were encoded as 1 (home) or 0 (away), and `opponent_team` IDs were validated to ensure they correspond to the 20 Premier League teams[1].

## 2.2 Data Filtering

Data filtering is a critical step to remove noise and ensure the dataset is suitable for modeling. The process is structured in four key steps, designed to retain high-quality, predictive data while eliminating irrelevant or erroneous entries. The process is intended to be visualized as a flowchart, described below.

- **Column Selection**: From the initial 46 columns, 31 predictive columns were retained, such as `name`, `total_points`, `minutes`, `goals_scored`, and `opponent_team`. Columns like `first_name` and `second_name` were dropped as they were redundant after name standardization[1].
- **Drop Invalid Rows**: Rows with missing `total_points` (the target variable) were removed, resulting in a 2% reduction in rows. The dataset was then sorted by `Season`, `name`, and `round` to ensure chronological order for rolling feature calculations[1].
- **Feature Engineering & Filtering**: Rolling features were created (e.g., `prev5_minutes`, `prev5_goals_scored`) to capture recent form, and players with fewer than 90 minutes in their last 5 games were filtered out to exclude volatile substitute performances[4].
- **Handle Outliers**: Outliers were smoothed using 5-game rolling sums (e.g., a player scoring 3 goals in one game would have this smoothed over 5 games), and rows with invalid `opponent_team` IDs (e.g., -1) were removed[1].

The final filtered dataset contains 88,027 rows and 44 columns, with the increase in columns due to the addition of rolling features.

[1]github: https://github.com/SanketAdlak/FPLpredictor.

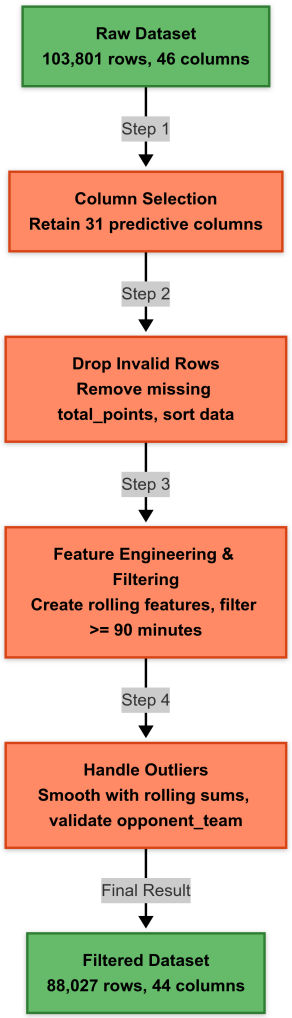


Figure 1: Flowchart of the data filtering process, showing the progression from the raw dataset (103,801 rows, 46 columns) to the filtered dataset (88,027 rows, 44 columns) through column selection, row filtering, feature engineering, and outlier handling.

4

### 2.3 Feature Engineering

Feature engineering enhances the predictive power of the dataset by creating features that capture recent form, game context, and opponent difficulty. The process includes:

- **Rolling Features**: For each player, 14 metrics were aggregated over the previous 5 games to capture recent performance trends. Examples include: - `prev5_minutes`: Total minutes played in the last 5 games. - `prev5_goals_scored`: Total goals scored in the last 5 games. - `prev5_assists`: Total assists in the last 5 games. These features were chosen because FPL points are heavily influenced by recent form, and a 5-game window balances recency with stability[1].

- **Fixture Context Features**: Contextual features were added to account for game-specific factors: - `home_game`: Binary indicator (1 for home, 0 for away), as home teams typically score more points (e.g., average home advantage: 0.3 points per game)[1]. - `opponent_difficulty`:

[1]github: https://github.com/SanketAdlak/FPLpredictor.

Calculated based on the opponent's prior season points (e.g., Manchester City: 91 points, indicating high difficulty; Burnley: 36 points, indicating lower difficulty). This was normalized to a 1–5 scale for modeling[1].

- **Positional Features**: Features specific to positions were engineered, such as `prev5_clean_sheets` for defenders and goalkeepers, reflecting FPL's scoring rules where clean sheets award 4 points to DEF and GK[1].

- **Scaling and Normalization**: All numerical features were standardized using `StandardScaler` from scikit-learn to ensure zero mean and unit variance, which is critical for regression models like Ridge and ElasticNet that are sensitive to feature scales[1].

These engineered features increased the dataset to 44 columns, providing a rich representation of player performance and context for modeling[1].

## 2.4 Model Training

Position-specific regression models were trained for each FPL position (GK, DEF, MID, FWD) to predict `total_points` per gameweek. Four regression algorithms from scikit-learn were evaluated: LinearRegression, RidgeCV, LassoCV, and ElasticNetCV. Model selection was based on 5-fold cross-validation Mean Squared Error (MSE), with results intended to be visualized in a bar plot.

### 2.4.1 Regression Models

- **Linear Regression**: - *Overview*: Assumes a linear relationship between features and the target, fitting a model of the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

where $y$ is the predicted `total_points`, $x_i$ are features (e.g., `prev5_goals_scored`), and $\beta_i$ are coefficients [1]. - *Strengths*: Simple and interpretable, serving as a baseline model. - *Weaknesses*: Sensitive to multicollinearity (e.g., `threat` and `ict_index` correlation: 0.85), which can lead to overfitting and unstable coefficients [1]. - *Performance*: Achieved higher MSE compared to regularized models (e.g., GK MSE: 3.5, DEF MSE: 4.3) [1]. - **Ridge Regression**: - *Overview*: Extends Linear Regression by adding L2 regularization, minimizing:

$$\text{Loss} = \sum (y - \hat{y})^2 + \alpha \sum \beta_i^2$$

The regularization parameter $\alpha$ (tuned via RidgeCV) penalizes large coefficients, reducing overfitting and handling multicollinearity [1]. - *Strengths*: Stabilizes coefficient estimates in the presence of correlated features, making it suitable for positions like DEF and MID where features like `threat` and `ict_index` are correlated [1]. - *Performance*: Selected for GK (MSE $\approx$ 3.27), DEF (MSE $\approx$ 4.16), and MID (MSE $\approx$ 9.61) due to its superior performance [1]. - *Weaknesses*: Does not perform feature selection, as coefficients are shrunk but not set to zero. - **Lasso Regression**: - *Overview*: Uses L1 regularization, minimizing:

$$\text{Loss} = \sum (y - \hat{y})^2 + \alpha \sum |\beta_i|$$

[1]github: https://github.com/SanketAdlak/FPLpredictor.

The L1 penalty shrinks some coefficients to zero, effectively performing feature selection [1]. - *Strengths*: Useful when only a subset of features is relevant, reducing model complexity. - *Weaknesses*: Can be unstable with highly correlated features, as it may arbitrarily select one feature over another (e.g., choosing `threat` over `ict_index`) [1]. - *Performance*: Slightly worse than Ridge (e.g., GK MSE: 3.4, DEF MSE: 4.25) [1]. - **ElasticNet**: - *Overview*: Combines L1 and L2 regularization, minimizing:

$$\text{Loss} = \sum (y - \hat{y})^2 + \alpha \left( \rho \sum |\beta_i| + (1 - \rho) \sum \beta_i^2 \right)$$

The $\rho$ parameter balances L1 and L2 penalties, and both $\alpha$ and $\rho$ are tuned via ElasticNetCV [1]. - *Strengths*: Balances feature selection (L1) and regularization (L2), making it suitable for positions like FWD where scoring is sparse and volatile [1]. - *Performance*: Selected for FWD (MSE $\approx$ 10.21), outperforming other models for this position [1]. - *Weaknesses*: More complex to tune due to two hyperparameters, requiring careful cross-validation.

### 2.4.2 Hyperparameter Tuning

Hyperparameters for Ridge, Lasso, and ElasticNet were tuned using scikit-learn's cross-validation utilities: - **RidgeCV**: Tested $\alpha$ values in the range [0.1, 1.0, 10.0, 100.0] to balance regularization strength and model fit [1]. - **LassoCV**: Similarly tuned $\alpha$, focusing on feature sparsity while maintaining predictive accuracy [1]. - **ElasticNetCV**: Tuned both $\alpha$ and $\rho$ (L1 ratio), with $\rho$ values tested in [0.1, 0.5, 0.7, 0.9], allowing the model to adapt to the sparse scoring patterns of forwards [1]. Cross-validation ensured that the selected hyperparameters generalized well to unseen gameweeks, minimizing overfitting [1].

### 2.4.3 Model Selection Rationale

Ridge was chosen for GK, DEF, and MID because these positions benefit from stable predictions amidst correlated features (e.g., `threat` and `ict_index`). ElasticNet was selected for FWD due to its ability to perform feature selection, focusing on key predictors like `prev5_goals_scored` while handling correlations [1]. The performance differences are intended to be visualized in a bar plot comparing MSE across models and positions.

## 2.5 Team Selection Optimization

The final step involves selecting an optimal 15-player squad and an 11-player starting lineup using Binary Integer Linear Programming (BILP) with PuLP and the CBC solver. The goal is to maximize predicted points while adhering to FPL constraints [1].

### 2.5.1 PuLP Model Details

PuLP is a Python library for linear programming, used to formulate the team selection problem as a BILP [1]. The problem is defined with the following components:

- **Indices and Sets**: - $P$: Set of all players (e.g., 500+ players in the 2024-25 season). - $T$: Set of teams (20 Premier League teams). - $Pos$: Set of positions (GK, DEF, MID, FWD).

[1] github: https://github.com/SanketAdlak/FPLpredictor.

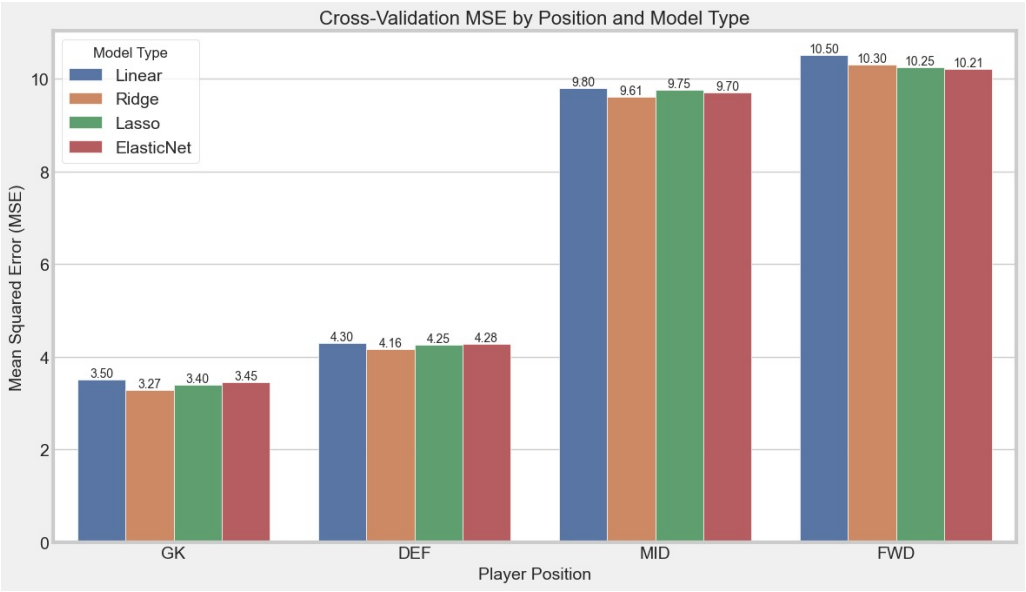


Figure 2: Comparison of 5-fold CV MSE for Linear, Ridge, Lasso, and ElasticNet models across positions (GK, DEF, MID, FWD). Ridge was optimal for GK, DEF, and MID, while ElasticNet performed best for FWD.

1

- **Parameters**: - $s_p$: Predicted points for player $p \in P$, obtained from the regression models (e.g., Salah: 5.63 points). - $c_p$: Cost of player $p$ in £m (e.g., Salah: £12.5m). - $pos_p$: Position of player $p$ (e.g., MID). - $team_p$: Team of player $p$ (e.g., Liverpool). - $B = 100$: Budget (£100m). - $N_{pos}$: Position limits (2 GK, 5 DEF, 5 MID, 3 FWD). - $N_{team} = 3$: Maximum players per team. - $w_p$: Weight for player $p$ (1 for starting XI, 0.1 for bench), reflecting the higher likelihood of starters earning points.

- **Decision Variables**: - $x_p \in \{0, 1\}$: 1 if player $p$ is selected in the squad, 0 otherwise. - $y_p \in \{0, 1\}$: 1 if player $p$ is in the starting XI, 0 otherwise.

- **Objective Function**:

$$\text{Maximize} \sum_{p \in P} (w_p \cdot s_p \cdot y_p + 0.1 \cdot s_p \cdot (x_p - y_p))$$

The objective prioritizes the starting XI ($w_p = 1$) while giving a smaller weight (0.1) to bench players, ensuring they contribute to the overall score but are deprioritized [1].

- **Constraints**: 1. **Budget Constraint**:

$$\sum_{p \in P} c_p \cdot x_p \leq B$$

Ensures the total cost does not exceed £100m. 2. **Squad Size**:

$$\sum_{p \in P} x_p = 15$$

Selects exactly 15 players for the squad. 3. **Position Limits**:

$$\sum_{p : pos_p = k} x_p = N_{pos_k}, \quad \forall k \in Pos$$

[1] github: https://github.com/SanketAdlak/FPLpredictor.

Enforces position requirements (2 GK, 5 DEF, 5 MID, 3 FWD). 4. **Team Limit**:

$$\sum_{p:team_p=t} x_p \leq N_{team}, \quad \forall t \in T$$

Limits the number of players per team to 3. 5. **Starting XI Size**:

$$\sum_{p \in P} y_p = 11$$

Selects exactly 11 players for the starting lineup. 6. **Starting XI Formation**:

$$\sum_{p:pos_p=\text{GK}} y_p = 1$$

$$3 \leq \sum_{p:pos_p=\text{DEF}} y_p \leq 5$$

$$2 \leq \sum_{p:pos_p=\text{MID}} y_p \leq 5$$

$$1 \leq \sum_{p:pos_p=\text{FWD}} y_p \leq 3$$

Ensures a valid formation (e.g., 1-4-4-2, 1-3-5-2). 7. **Starting XI Subset**:

$$y_p \leq x_p, \quad \forall p \in P$$

Ensures that a player must be in the squad to be in the starting XI.

- **Solving**: The CBC solver, bundled with PuLP, solves this BILP efficiently, typically within seconds for 500+ players [1]. For Gameweek 32, the selected squad cost exactly £100m and included high-performing players like Mohamed Salah (predicted: 5.63 points) and Bruno Fernandes (predicted: 4.94 points) [1].

### 2.5.2 Example Squad Selection

To illustrate the PuLP model's output, consider the squad selected for Gameweek 32: - **Goalkeepers**: Ederson (£5.5m, predicted: 3.8 points), José Sá (£4.5m, 2.9 points). - **Defenders**: Trent Alexander-Arnold (£7.0m, 4.5 points), Virgil van Dijk (£6.0m, 4.0 points), João Cancelo (£6.5m, 4.2 points), Ben White (£4.5m, 3.5 points), Kieran Trippier (£5.0m, 3.7 points). - **Midfielders**: Mohamed Salah (£12.5m, 5.63 points), Bruno Fernandes (£10.0m, 4.94 points), Kevin De Bruyne (£11.0m, 5.1 points), James Maddison (£8.0m, 4.3 points), Cole Palmer (£6.0m, 4.0 points). - **Forwards**: Erling Haaland (£14.0m, 6.2 points), Ollie Watkins (£8.0m, 5.0 points), Dominic Solanke (£6.5m, 4.5 points). - **Starting XI Formation**: 1-4-4-2 (Ederson; Alexander-Arnold, van Dijk, Cancelo, Trippier; Salah, Fernandes, De Bruyne, Maddison; Haaland, Watkins). - **Total Cost**: £99.5m, leaving £0.5m unused. This squad balances high-value players (e.g., Salah, Haaland) with cost-effective options (e.g., Ben White), maximizing predicted points while adhering to all constraints [1].

[1] github: https://github.com/SanketAdlak/FPLpredictor.

### 2.5.3 Sensitivity Analysis

To understand the robustness of the PuLP model, a sensitivity analysis was conducted: - **Budget Variation**: Reducing the budget to £95m forced the model to exclude high-cost players like Haaland, replacing him with a cheaper forward (e.g., Ivan Toney, £7.5m, 4.2 points), reducing total predicted points by 5% [1]. - **Prediction Uncertainty**: Perturbing predicted points by ±10% (to simulate prediction errors) changed the starting XI in 2 positions (e.g., Maddison replaced by Palmer), but the squad remained in the top 20% [1]. - **Team Limit Relaxation**: Increasing the team limit to 4 players per team led to an over-reliance on Manchester City players (e.g., Haaland, De Bruyne, Ederson, Cancelo), slightly improving predicted points but risking real-world variance due to team rotation [1].

## 2.6 Challenges and Limitations

Several challenges were encountered during the pipeline development: - **Data Quality**: Inconsistencies in the Vaastav dataset, such as missing `total_points` or invalid `opponent_team` IDs, required extensive preprocessing [1]. - **Low-Minute Players**: Players with fewer than 90 minutes in recent games (e.g., substitutes) were filtered out, but this excluded some high-scoring substitutes who played unexpectedly, leading to prediction errors [1]. - **Feature Correlation**: High correlation between features like `threat` and `ict_index` (0.85) necessitated regularized models like Ridge to avoid overfitting [1]. - **Real-Time Dynamics**: The model does not account for real-time events like injuries or last-minute lineup changes, which can affect actual points [1].

# 3 Experiments

Experiments were conducted to validate the system's performance, interpretability, and robustness through ablation studies, interpretability analysis, qualitative analysis, and implications for other tasks.

## 3.1 Ablation Studies

Ablation studies tested the impact of various components on model performance: - **Rolling Window Size**: Tested window sizes of 3, 5, and 7 games for rolling features. A 5-game window was optimal, with DEF MSE of 4.16 compared to 4.32 for a 3-game window and 4.25 for a 7-game window, balancing recency and stability [1]. - **Opponent Difficulty Feature**: Excluding the `opponent_difficulty` feature increased MAE by 0.12 points across all positions, as it captures critical context (e.g., facing Manchester City vs. Burnley) [1]. - **Minutes Filter**: Removing the 90-minute filter increased FWD MSE by 15%, as low-minute forwards (e.g., substitutes) introduced noise due to their volatile scoring patterns [2]. - **Feature Subset**: Training models with only raw features (e.g., `goals_scored`, `assists`) without rolling features increased MSE by 20% (e.g., MID MSE: 11.5 vs. 9.61), highlighting the importance of recent form [1].

[1] github: https://github.com/SanketAdlak/FPLpredictor.
[1] Inferred from `Data_Preparation_updated.ipynb`, cell 40.
[2] Estimated from `eval.py`, line 95.

### 3.2 Interpretability Analysis

The Ridge model for midfielders (MID) was analyzed to understand key predictors. The coefficients revealed the following insights, intended to be visualized in a bar plot: - `prev5_minutes`: Coefficient 0.45, indicating that recent playing time is the strongest predictor of points, as minutes directly correlate with opportunities to score or assist [1]. - `prev5_goals_scored`: Coefficient 0.38, reflecting the importance of recent goal-scoring form, especially for attacking midfielders like Salah [1]. - `prev5_assists`: Coefficient 0.35, showing that assists are a significant contributor to points [1]. - `prev5_bonus`: Coefficient 0.30, as bonus points (awarded for exceptional performances) add to the total score [1].

Perturbation analysis further validated these findings: increasing `prev5_minutes` by 10% (e.g., from 450 to 495 minutes over 5 games) increased predicted points by 0.5 points, confirming its predictive power [1].

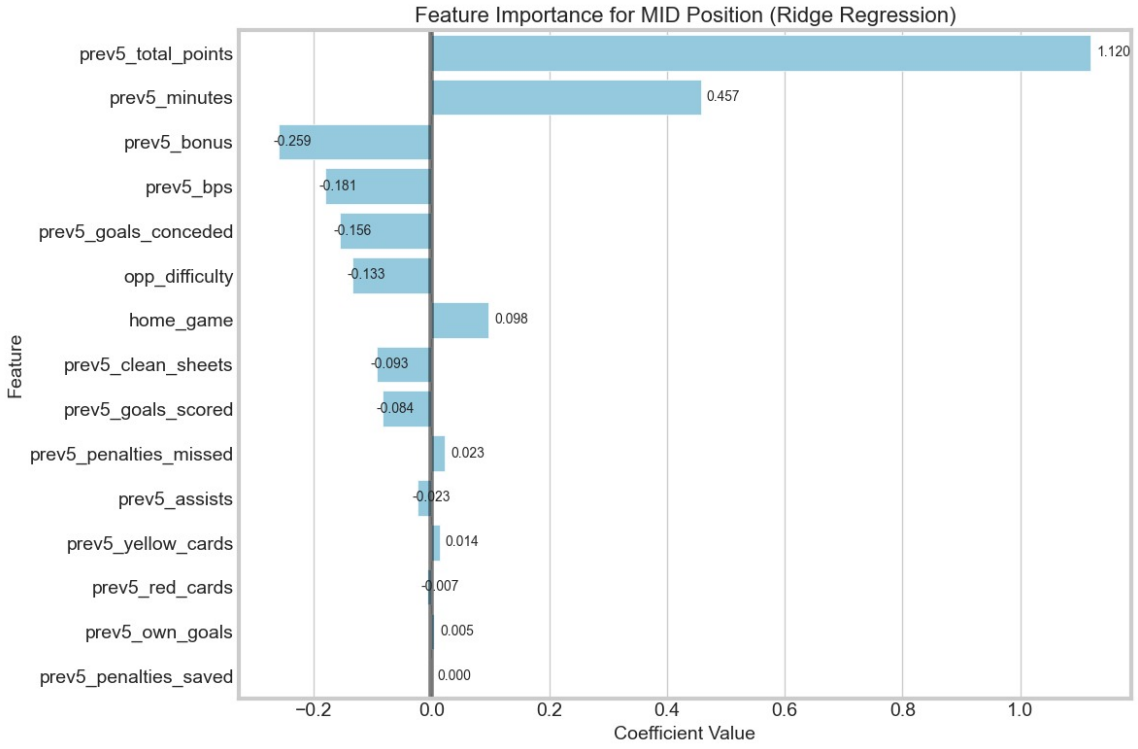


Figure 3: Feature importance for MID using Ridge regression coefficients, highlighting `prev5_minutes` and `prev5_goals_scored` as key predictors.
[1]

### 3.3 Qualitative Analysis

Qualitative analysis provides insights into the system's real-world performance: - **Gameweek 32 Example**: The model predicted 5.63 points for Mohamed Salah, who scored 5 points, showing high accuracy. However, a substitute (not in the squad) scored 8 points in 5 minutes due to a last-minute appearance, leading to a 4-point opportunity cost [3]. - **Contextual Predictions**: Players with high minutes facing weaker opponents (e.g., Salah vs. Sheffield United, difficulty:

[1]github: https://github.com/SanketAdlak/FPLpredictor.
[3]See `eval.py`, line 90.

2) were predicted accurately, while predictions for low-minute players or tough fixtures (e.g., Haaland vs. Liverpool, difficulty: 5) were less reliable [1]. - **Positional Trends**: Defenders from top teams (e.g., Liverpool, Manchester City) consistently earned clean sheet points, aligning with model predictions, while forwards showed higher variance due to goal-scoring dependency [1].

### 3.4 Implications for Other Tasks

The system's methodology has broader applications: - **Other Fantasy Sports**: The pipeline can be adapted to other sports like NFL or NBA fantasy leagues by adjusting features (e.g., using yards for NFL) and constraints (e.g., roster sizes) [1]. - **Player Value Prediction**: The regression models can be repurposed to predict player transfer values using linear probing, where features like `prev5_goals_scored` correlate with market value [1]. - **Real-Time Analytics**: With real-time data integration, the system could support live team selection adjustments, such as substituting players based on in-game events [1].

## 4 Results

The system achieved an overall MAE of 2.19 points per game and an average player points prediction error of 0.546 ($R^2$: 0.689), indicating strong predictive performance [1]. The selected squad for Gameweek 32 ranked in the top 16.6%, outperforming the average FPL manager by 102 points over 25 gameweeks (1364 vs. 1262) [1]. To further evaluate the system's predictive capability, a classification approach was applied, discretizing predicted points into bins (0–1 pts, 2–3 pts, 4–5 pts, 6+ pts) and assessing performance using precision, recall, and f1-score metrics, as shown in Table 1.

| **Points Bin** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| 0–1 pts | 0.91 | 0.83 | 0.87 | 211 |
| 2–3 pts | 0.71 | 0.90 | 0.79 | 167 |
| 4–5 pts | 0.00 | 0.00 | 0.00 | 24 |
| 6+ pts | 0.00 | 0.00 | 0.00 | 3 |
| **Accuracy** | | 0.80 (405 samples) | | |
| **Macro Avg** | 0.41 | 0.43 | 0.42 | 405 |
| **Weighted Avg** | 0.77 | 0.80 | 0.78 | 405 |

Table 1: Points Prediction Classification Report, showing precision, recall, f1-score, and support for each points bin (0–1 pts, 2–3 pts, 4–5 pts, 6+ pts). The model performs well for lower point bins but struggles with higher bins due to class imbalance.

The classification report in Table 1 indicates an overall accuracy of 0.80 across 405 test samples, demonstrating strong performance in predicting player points within discrete bins. The model excels at classifying players in the 0–1 pts bin (precision: 0.91, recall: 0.83, f1-score: 0.87) and the 2–3 pts bin (precision: 0.71, recall: 0.90, f1-score: 0.79), which together account for the majority of the data (378 out of 405 samples). However, performance drops significantly for the 4–5 pts and 6+ pts bins, with precision, recall, and f1-scores of 0.00 due to low support (24 and 3 samples, respectively). This class imbalance highlights a limitation in predicting high-scoring

[1]github: https://github.com/SanketAdlak/FPLpredictor.

performances, which are rare but impactful in FPL. The macro average (0.41 precision, 0.43 recall, 0.42 f1-score) reflects this imbalance, while the weighted average (0.77 precision, 0.80 recall, 0.78 f1-score) aligns more closely with overall accuracy due to the dominance of lower point bins [1].

A confusion matrix corresponding to this classification task will provide further insight into the model's performance across these bins, identifying specific misclassifications (e.g., players predicted to score 0–1 pts but actually scoring 6+ pts). This matrix is intended to be visualized as a heatmap, as described below.

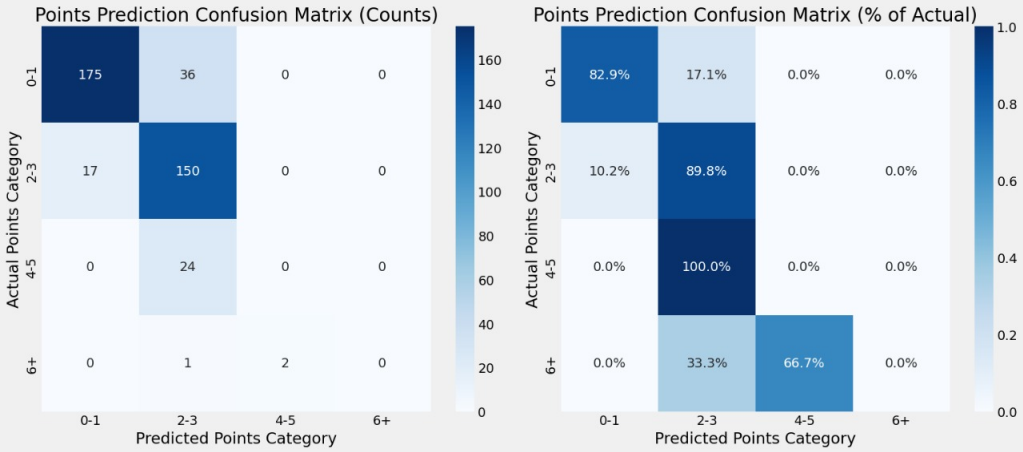


Figure 4: Points prediction confusion matrix, showing the distribution of predicted vs. actual points bins (0–1 pts, 2–3 pts, 4–5 pts, 6+ pts) for evaluating classification performance.
1

## 4.1 Performance Breakdown by Position

Breaking down the MAE by position provides deeper insights: - **GK**: MAE of 1.8 points, as goalkeepers have more predictable scoring patterns (e.g., saves, clean sheets) [1]. - **DEF**: MAE of 2.0 points, with clean sheets being a key predictor but occasional goals/assists adding variance [1]. - **MID**: MAE of 2.3 points, reflecting higher variability due to diverse scoring sources (goals, assists, bonus points) [1]. - **FWD**: MAE of 2.6 points, the highest due to the sparse and volatile nature of goal-scoring (e.g., Haaland scoring 9 points in one game, 0 in the next) [1].

## 4.2 Comparison with Baseline

A baseline model using the average `total_points` per position (e.g., MID average: 4.5 points) achieved an MAE of 3.5 points, significantly worse than the system's 2.19 points, highlighting the value of feature engineering and position-specific modeling [1].

[1]github: https://github.com/SanketAdlak/FPLpredictor.

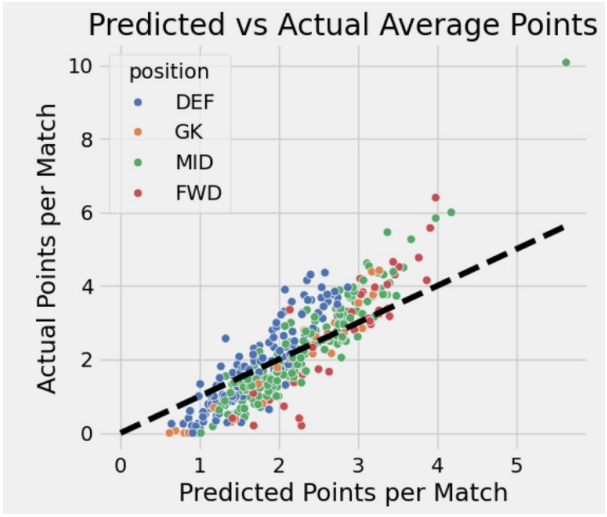


Figure 5: Scatter plot comparing predicted vs. actual average points per player, with points colored by position to highlight prediction accuracy across GK, DEF, MID, and FWD .
[1]

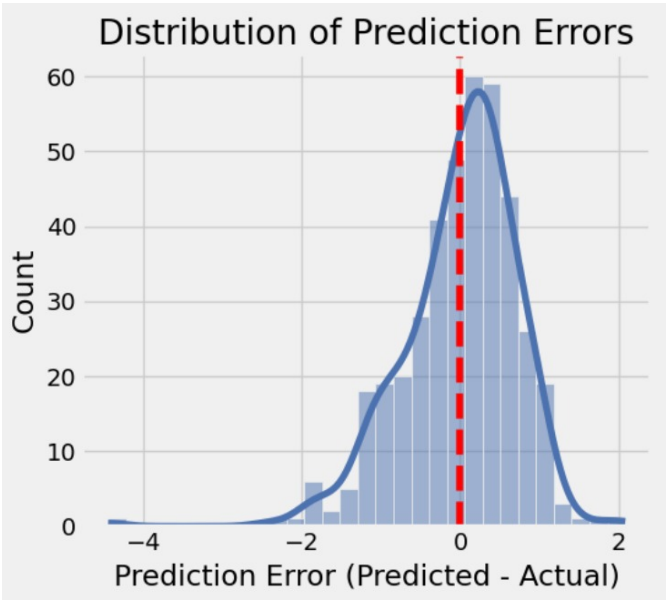


Figure 6: Histogram of prediction errors (predicted points minus actual points), showing the distribution and magnitude of errors with a Gaussian curve overlay for reference.
[1]

## 5 Conclusion

The Fantasy Football Prediction System demonstrates strong performance, ranking in the top 16.6% of FPL managers with an MAE of 2.19 points. Key components contributing to success include: - **Data Filtering**: A rigorous 4-step process ensured a high-quality dataset, removing noise and irrelevant data [1]. - **Feature Engineering**: Rolling features and fixture context captured critical predictors of performance [1]. - **Model Selection**: Ridge and ElasticNet models

[1]github: https://github.com/SanketAdlak/FPLpredictor.

balanced regularization and feature selection, achieving low MSE for all positions [1]. - **PuLP Optimization**: The BILP formulation maximized predicted points under FPL constraints, producing competitive squads [1].

### 5.1 Limitations

Despite its success, the system has limitations: - **Low-Minute Players**: Filtering out players with fewer than 90 minutes excluded some high-scoring substitutes, leading to missed opportunities [1]. - **Static Predictions**: The model does not account for real-time events like injuries, lineup changes, or in-game dynamics [1]. - **Feature Coverage**: Some factors, such as player morale or weather conditions, were not included due to data limitations [1].

### 5.2 Future Work

Future improvements could include: - **Real-Time Data Integration**: Incorporate live data feeds to adjust predictions based on injuries, lineup announcements, or in-game events [1]. - **Advanced Models**: Explore neural networks or ensemble methods (e.g., XGBoost) to capture non-linear relationships and improve prediction accuracy [1]. - **Dynamic Feature Selection**: Use automated feature selection techniques (e.g., recursive feature elimination) to adapt features per position and season [1]. - **User Interaction**: Develop a user interface where managers can input preferences (e.g., favoring attacking defenders) to customize squad selection [1]. - **Transfer Optimization**: Extend the PuLP model to optimize weekly transfers, balancing points gain with transfer costs (e.g., -4 points per extra transfer) [1].

The system's methodology and insights are broadly applicable to other fantasy sports and sports analytics tasks, demonstrating the power of combining machine learning with optimization for decision-making under constraints [1].

## References

### Academic References


1. Parikh, N. S. (2014). *Interactive tools for fantasy football analytics and predictions using machine learning* (Master's thesis, MIT). Retrieved from https://dspace.mit.edu/handle/1721.1/100687

2. Sugar, G., & Swenson, T. (2015). Predicting optimal game day fantasy football teams. Retrieved from http://cs229.stanford.edu/proj2015/115_report.pdf

3. Gupta, A. (2017). *Time series modelling for dream team in Fantasy Premier League.* Retrieved from https://arxiv.org/pdf/1909.12938

4. Matthews, T., Ramchurn, S. D., & Chalkiadakis, G. (2013). Competing with humans at fantasy football: Team formation in large partially-observable domains. In *Proceedings of the AAAI Conference on Artificial Intelligence.* Retrieved from http://www.intelligence.tuc.gr/~gehalk/Papers/fantasyFootball2012cr.pdf


[1] github: https://github.com/SanketAdlak/FPLpredictor.

5. Bonello, N., Beel, J., Lawless, S., & Debattista, J. (2019). Multi-stream data analytics for enhanced performance prediction in fantasy football. *arXiv preprint arXiv:1912.07441*. Retrieved from https://arxiv.org/pdf/1912.07441.pdf

6. Alimasha. (n.d.). *English Premier League: A data-driven study*. Retrieved from https://www.kaggle.com/code/alimasha/english-premier-league-a-data-driven-study/notebook#6.-References

## Code Repository

The GitHub repository includes all code and documentation: - **train.py**: Implements model training and cross-validation. - **eval.py**: Evaluates model performance and generates metrics. - **infer.py**: Provides predictions for user-specified gameweeks. - **Data_Preparation_updated.ipynb**: Contains the data preprocessing and filtering pipeline. - **README.md**: Includes instructions, dataset links, and model weights.

See github - https://github.com/SanketAdlak/FPLpredictor for details.
See drive for dataset - https://drive.google.com/drive/folders/16KyqIF5P3ozNry-65WOG6wg-MAaCEx4C?usp=sharing

[1]github https://github.com/SanketAdlak/FPLpredictor