Player Performance Evaluation Using Support Vector Machines Across Multiple Sports: A Comparative Machine Learning Approach

Surya Pramod Bharadwaja Josyula, [1] Vijaynath Kommaraju, [1] Mamatha T. M [2]

[1] *Department of School of Computing, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Kasavanahalli, Bengaluru 560035 , Karnataka, India*

[2] *Department of Mathematics, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Kasavanahalli, Bengaluru 560035, Karnataka, India*

bl.sc.u4aie24252@bl.students.amrita.edu
bl.sc.u4aie24233@bl.students.amrita.edu
tm_mamatha@blr.amrita.edu

Abstract

This research paper presents a machine learning methodology for assessing player performance in three major sports: football, cricket, and basketball, utilizing Support Vector Machines (SVM). The model converts performance metrics into classes (Low, Medium, High) through percentile-based cutoffs, thereby ensuring balanced classification. Each sport utilizes specific mathematical formulations to calculate an overall impact score based on essential performance attributes (e.g., batting average, wickets, strike rate for cricket; goals, assists, tackles for football; points, rebounds, assists for basketball). The datasets, downloaded from KaggleHub, undergo standardized preprocessing, feature scaling, and model evaluation. This approach utilizes statistical modeling combined with machine learning to establish a comprehensive, data-informed framework for evaluating performance. The findings indicate a high level of accuracy: 98.98% for cricket, 98.77% for football, and 98.74% for basketball - demonstrating the dependability of the SVM across different sports, thereby offering prospects for player performance evaluation in sports analytics.

Keywords: Support Vector Machine (SVM), Sports Analytics, Performance Classification, Machine Learning, Data Normalization, Player Impact Modelling.

1. Introduction

The increasing availability of high-resolution sports data has transformed the landscape of performance analytics across major global games such as cricket, football, and basketball. Modern competitive environments in the Indian Premier League (IPL), international football leagues, and the National Basketball Association (NBA) demand objective, consistent, and data-centric methods for evaluating player performance. Traditional assessment techniques often rely on selective statistics—such as runs or wickets in cricket, goals or assists in football, and points or rebounds in basketball—which, when considered independently, fail to capture the multidimensional skill set required for optimal overall performance. Furthermore, qualitative judgements or manual comparisons introduce subjectivity, making cross-player and cross-sport evaluation inconsistent. To overcome these limitations, this study proposes a unified analytical framework capable of integrating diverse performance indicators across all three sports. The framework employs metric normalization, statistical aggregation, and machine learning classification—specifically, a Support Vector Machine (SVM) model—to categorize players into performance tiers and generate a composite final score. This standardized approach allows each player to be evaluated using a balanced combination of domain-specific metrics and predictive modelling, enabling deeper insights into their relative strengths. In addition, sport-specific role detection, score decomposition, and visual analytics (including distribution plots and ranking animations) are incorporated to enhance interpretability and support strategic decision-making. By consolidating these techniques, the study provides a robust, scalable, and transparent methodology for comprehensive performance evaluation across cricket, football, and basketball.

2. Literature Survey

1. [1] R. Kumar and A. Singh, support vector machine-based cricket talent identification model (SVMCTI).
   Developed an SVM model to identify cricket talent objectively using player data (physical, technical, psychological). The model selected key features and trained on labeled data to predict new players' talent levels. Found to be more accurate and less biased than human selection.

2. [2] T. Wong and H. Chen, support vector machine-based prediction system for football match result.
   Used historical EPL (2014–15) match data to predict outcomes via SVM with a Gaussian kernel. Preprocessed data (cleaning, normalization, encoding). Achieved 53.3% accuracy—moderate but demonstrated SVM's potential for sports result prediction.

3. [3] V. N. Vapnik and C. Cortes, support vector machines for classification and regression.
   Explained SVM principles using simulated and chemical datasets. Applied linear and RBF kernels, soft margins, and PCA for preprocessing. Showed SVMs' high accuracy for both classification and regression when parameters are well-tuned.

4. [4] Y. Li, Q. Zhang and S. Wang, forecasting sports injuries with machine learning.
   Used player performance, medical, and wearable data to predict injuries using an RBF-kernel SVM. Preprocessed with normalization and feature selection. Found SVM more accurate than Random Forest or Logistic Regression, helping early injury prevention.

5. [5] M. Awad and R. Khanna, efficient learning machines: theories, concepts, and applications for engineers and system designers.
   Past studies review classical ML models like SVMs and decision trees, along with modern deep learning methods such as autoencoders and DBNs. Research also highlights preprocessing, feature extraction, and evaluation metrics as essential for performance. Temporal models and bio-inspired approaches like HMMs and HTM offer additional methods for sequence learning.

6. [6] P. Y. Lakshmi, S. Sanjaykumar, M. Dharuman and A. Elangovan, using support vector regression kernel models for cricket performance prediction in the women's premier league 2024.
   Past studies in cricket analytics have applied machine-learning models to understand player performance, emphasizing methods such as SVR, statistical analysis, and kernel-based learning. Researchers highlight how preprocessing steps—including imputation, outlier removal, and data normalization—significantly improve the reliability of predictions. Comparative evaluations of ANOVA, Bessel, and radial basis kernels show that different mathematical functions capture variance, periodic patterns, and nonlinear relationships in cricket data. Recent work also demonstrates that advanced kernel approaches can outperform traditional statistical techniques in modelling complex player dynamics.

7. [7] S. Han, Q. Cao and M. Han, parameter selection in SVM with RBF kernel function.
   Past studies examining SVM performance emphasize the importance of optimal parameter selection, particularly for the RBF kernel, which heavily influences classification accuracy. Researchers have explored methods such as grid search, double-linear search, and improved hybrid techniques to tune parameters like C and γ efficiently. These works highlight how kernel choice and parameter control directly affect model complexity, error rates, and generalization capability across different datasets.

8. [8] S. Suthaharan, machine learning models and algorithms for big data classification.
   Past studies on big data classification discuss a wide range of machine learning models, including decision trees, random forests, SVMs, and deep learning architectures designed for large-scale datasets. Researchers emphasize the role of distributed systems like Hadoop and MapReduce in enabling scalable data processing and efficient model training. The book also highlights representation learning, dimensionality reduction, and hierarchical and layered models as essential components for handling high-volume, high-velocity data.

9. [9] H. Liu, technology-based sports performance enhancement: a conference review.
   Past studies in sports technology highlight how AI-driven wearables, computer vision systems, and machine-learning models enhance athletic monitoring and performance analysis. Research demonstrates that real-time physiological tracking, automated movement detection, and data-driven tactical analysis provide deeper insights into athlete behavior and training optimization. Innovations such as ECG-based monitoring, deep-learning jump-measurement systems, and unsupervised clustering for match analysis have significantly advanced performance evaluation and coaching efficiency.

10. [10] T. Evgeniou and M. Pontil, workshop on support vector machines: theory and applications.
    Past studies highlight how Support Vector Machines emerged from statistical learning theory, emphasizing core ideas like structural risk minimization, VC dimension, and

kernel-based hypothesis spaces. Research has shown SVMs to be effective across applications such as medical diagnosis, time-series prediction, and face authentication, with various kernel designs improving adaptability to different tasks. Methods such as primal–dual optimization, decomposition algorithms, and local SVM models further enhance training efficiency and predictive performance.

2.1 Research Gaps

1. [1] R. Kumar and A. Singh, support vector machine-based cricket talent identification model (SVMCTI).
   While the model objectively identifies cricket talent, it mainly focuses on limited player features. It does not fully integrate psychological or contextual factors, and real-world validation across diverse datasets is lacking.

2. [2] T. Wong and H. Chen, support vector machine-based prediction system for football match result.
   The study achieved only moderate accuracy (53.3%) due to a small dataset and limited training examples. It did not explore advanced feature selection, larger datasets, or hybrid models to improve predictive performance.

3. [3] V. N. Vapnik and C. Cortes, support vector machines for classification and regression.
   Although the paper explains SVM theory and tuning, it focuses mainly on chemical datasets. Its findings are not directly applied to sports analytics, leaving a gap in how SVM methods can be optimized for complex, real-world sports data.

4. [4] Y. Li, Q. Zhang and S. Wang, forecasting sports injuries with machine learning.
   4.1 The model predicts injury risk effectively but focuses on a single outcome, lacking integration of multi-dimensional data (technical, psychological, tactical) and thus limiting holistic performance evaluation.
   4.2 Contributions of this paper

   4.3 Unified SVM-Based Framework: Developed a common machine learning pipeline applicable across multiple sports domains—football, cricket, and basketball—using a Support Vector Classifier (SVC) with RBF kernel.
   4.4 Mathematical Modeling of Player Impact: Proposed sport-specific performance metrics that integrate offensive, defensive, and efficiency statistics into a single "overall impact" measure.
   4.5 Visualization for Interpretability: Introduced dynamic leaderboard animations to visually compare player rankings, enhancing interpretability for both analysts and non-technical stakeholders.

5. [5] M. Awad and R. Khanna, efficient learning machines: theories, concepts, and applications for engineers and system designers.
   Current models still lack scalability for fast, real-world data and remain sensitive to noise in temporal tasks. Bio-inspired methods are under-validated and need stronger

benchmarking. Deep models also face challenges in training cost, interpretability, and deployment efficiency.

6. [6] P. Y. Lakshmi, S. Sanjaykumar, M. Dharuman and A. Elangovan, using support vector regression kernel models for cricket performance prediction in the women's premier league 2024.
Existing studies using SVR kernels in cricket prediction rely on limited performance features and do not include broader factors like match conditions or player fitness. Current models are mostly single-method approaches, leaving room for ensemble or hybrid techniques. Research also lacks testing across multiple seasons or leagues, which limits generalizability of the predictions.

7. [7] S. Han, Q. Cao and M. Han, parameter selection in SVM with RBF kernel function.
Despite effective tuning strategies, existing methods still face limitations such as high computational cost in grid search and sensitivity to initial parameter selection in linear methods. Current approaches also lack theoretical guidance for choosing optimal parameter ranges and struggle with scalability for large datasets. Further research is needed to develop faster, adaptive, and more automated parameter-selection frameworks that maintain accuracy while reducing computational load.

8. [8] S. Suthaharan, machine learning models and algorithms for big data classification.
Despite significant progress, existing work still faces limitations in scalability when data grows beyond distributed system capacity. Current models struggle with high-dimensional feature spaces, requiring better automated feature engineering and reduction techniques. The literature also notes a gap in developing faster, more adaptive learning algorithms that maintain accuracy while reducing computational cost across big data environments.

9. [9] H. Liu, technology-based sports performance enhancement: a conference review.
Despite these advancements, current technologies still face limitations in data accuracy, sensor reliability, and real-world validation across diverse sports environments. Machine-learning models struggle with real-time decision support and limited interpretability, while psychological assessment tools like PFT lack ecological validity outside controlled settings. The literature also underscores gaps in teacher–coach integration, long-term impact evaluation, and scalable implementation of tech-based interventions.

10. [10] T. Evgeniou and M. Pontil, workshop on support vector machines: theory and applications.
Despite these advances, existing SVM approaches face challenges in scaling to very large datasets due to computationally heavy quadratic programming. Kernel and parameter selection still lack automated, theoretically guided solutions, and imbalance handling or class-specific tuning remains limited. Additionally, while local SVMs and kernel variations show promise, broader evaluation across diverse data types is needed to improve generalization and practical applicability.

2.2 Motivation

Traditionally, player evaluation in cricket, football, and basketball are conducted using isolated statistics or subjective expert judgments, which do not account for the multifaceted nature of modern sports performance, and in the age of data explosion in leagues like the IPL, FIFA competitions, and the NBA, there is an urgent requirement for a common objective, data-driven framework for evaluating athletes across different roles and sporting contexts. Existing models only address a single sport or a single dimension of performance, and are not able to be compared across the different sports and sporting contexts. The purpose of this project is to develop a unified framework for statistical normalization, machine learning classification, and role-specific analytics to enable accurate ranking and decision-making that are applicable to a wide range of sports and sporting contexts.

2.3 Contributions

This project makes several significant contributions toward the development of a unified and data-driven sports analytics framework. First, it introduces a standardized methodology capable of evaluating player performance across three distinct sports—cricket, football, and basketball—addressing the limitations of sport-specific or single-metric evaluation systems. Second, the study proposes a hybrid scoring model that combines normalized statistical metrics with Support Vector Machine (SVM)–based classification, enabling a more consistent and objective assessment of player performance tiers. Third, the framework incorporates role-aware performance analysis, allowing accurate ranking of batsmen, bowlers, wicketkeepers, football positional players, and basketball role groups. Finally, the system enhances interpretability through visual analytics such as distribution graphs, confusion matrices, and animated ranking charts, making the results intuitive and accessible for analysts, coaches, and researchers.

3. Novelty

The novelty of this project lies in the development of a unified, cross-sport performance evaluation framework that integrates data from cricket (IPL), football, and basketball (NBA) using a consistent analytical structure. Unlike traditional studies that focus on a single sport or rely on isolated performance indicators, this project introduces a multi-domain scoring methodology that normalizes heterogeneous metrics, inverts sport-specific parameters, and generates a balanced performance index applicable across different game formats.

A key contribution is the incorporation of a hybrid scoring architecture that combines statistical normalization with machine learning–based classification. The utilization of a Support Vector Machine (SVM) classifier to categorize players into performance classes - followed by a weighted fusion of predicted class scores and statistical performance scores - provides a more robust, consistent, and interpretable evaluation system compared to conventional single-metric or rule-based models.

Furthermore, the project presents a role-aware evaluation mechanism for each sport, enabling specialized ranking of batsmen, bowlers, wicketkeepers, football attackers/midfielders/defenders, and basketball guards/forwards/centers. This role-based decomposition ensures higher granularity and improves the accuracy of player comparisons within and across roles.

Finally, the inclusion of advanced visualization tools, such as distribution analysis, heatmaps, and animated ranking charts, enhances transparency and interpretability—making the framework not only analytically strong but also visually intuitive for analysts, coaches, and researchers. Collectively, these contributions establish a novel, scalable, and cross-sport methodology for comprehensive player performance assessment.

4. Mathematical Modeling

4.1 Football Player Performance Analysis

In this methodology, a player's overall performance is assessed through an SVM based modelling approach to convert raw player data into a unified performance score. The method involves cleaning and standardizing the data, normalizing all attributes to a common scale, constructing a performance measure and classifying players. The overall modelling process included the following steps:

- Normalization of raw metrics:

  All metrics are scaled down between 0 and 1:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

  Each player's performance score is the mean of normalized features.

$$Performance\ Score = \left(\frac{1}{n}\sum_{i=1}^{n} X_{norm,i}\right)\times 100$$

- Percentile based Grouping:

  Players are grouped into Low, Medium, High classes based on their performance scores.

- SVM Classification (RBF Kernel):

  The Support Vector Machine classifies players into Low, Medium, High based on modelling decisions

$$\hat{y} = SVM(Standardize(X))$$

  RBF Kernel is used to capture the non-linear relationships between the parameters.

- Final Score Calculation:

  The final score of a player is calculated using:

$$Final\ Score = 0.5\times Performance\ Score + 0.5\times Predicted\ Class\ Score$$

  Where class score is mapped as:

  Low = 0, Medium = 50, High = 100

4.2 Cricket Player Performance Analysis

The analysis used SVM-based modelling which is used to convert the given player data into performance scores. This method involves cleaning the data, bringing all metrics to a common scale, combining them into a single performance measure and finally classifying players on the basis of these scores. The process included the following steps:

- Data Preparation and Analysis:

All values were cleaned and scaled between 0 and 1 using the formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Bowling metrics, where lower is better (Economy, Bowling SR) are inverted using:

$$x_{inv} = x_{max} - x$$

- Player Performance calculation:

A performance score is calculated for a player by averaging all normalized batting and bowling parameters

$$Performance\ score = \left(\frac{\sum x'}{6}\right) \times 100$$

- SVM rating and final score:

Each player is grouped into 3 classes: Low, Medium and High based on their performance scores. SVM predicts the class again and the final score is calculated

$$Final\ score = 0.7(performance) + 0.3(SVM\ score)$$

- Role classification and ranking:

Roles for each player (Batsman, Bowler, All-rounder, Wicketkeeper) are assigned based on which score is highest. Players are ranked based on the final score.

4.3 Basketball Player performance analysis

The analysis employs a modelling approach based on SVM to transform NBA player statistics into a consolidated performance score. The initial step involves cleaning the dataset and standardizing all six essential basketball metrics to a uniform numerical scale. These normalized features are subsequently aggregated to produce a statistical performance score for each player. With these scores, players are categorized into Low, Medium, and High tiers, followed by training an SVM classifier to enhance these classifications through non-linear decision boundaries. Ultimately, the statistical and predicted scores are merged to derive a final ranking score. The entire modelling process comprises the following steps:

- Normalization of parameters:
  All parameters are normalized in the range of 0-1 using:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

The overall performance score is calculated by

$$Performance\ Score = (\frac{1}{6}\sum_{i=1}^{6} X_{norm,i})\times 100$$

- Percentile based classification:

  Players are divided into Low, Medium, and High classes based on performance score:

$$Low = X \le Q_{33} Medium = Q_{33} < X \le Q_{66} High = X > Q_{66}$$

  where Q_{33} and Q_{66} are the 33rd and 66th percentiles.

- SVM classification with RBF Kernel:

  The model uses SVM to classify players into Low/Medium/High classes and RBF kernel is applied on non-linear boundaries:

$$K\left(x_i, x_j\right) = e^{-\gamma\|x_i - x_j\|^2}$$

  The SVM classifier works on standardized data:

$$X_{scaled} = \frac{X-\mu}{\sigma}$$

- Final score calculation:

  A final score is calculated to classify players:

  Final Score=0.7×Performance Score+0.3×Predicted Class Score

5. Methodology

5.1 Football Player Performance Analysis

- Data Cleaning and preparation

  The FIFA dataset is loaded, and all performance metrics (goals, assists, dribbles, tackles, etc.) are cleaned by converting missing or non-numerical values to valid numbers. This ensures data integrity.

- Performance Scoring

  All metrics are normalized and performance score is calculated for each player.

- SVM classification

  A SVM model is trained to classify players according to classes. A combined final score is calculated using statistical performance score and SVM predicted class to improve accuracy.

- Visualization and ranking

Graphs are plotted for each metric and animated bar charts show the Top-10 players in each position and Top-10 overall players. Finally, the script prints ranking tables showing the best performers.

5.2 Cricket Player Performance Analysis

- Data Cleaning and Normalization:
  The IPL dataset is loaded and all performance-related columns (batting, bowling, keeping) are cleaned by converting text values, missing data, and numeric inconsistencies into usable numbers.

- Performance Score Calculation

  Batting, bowling, and overall performance metrics are normalized using MinMaxScaler to bring them into a 0–1 range. An overall performance score (0–100) is calculated by averaging all normalized metrics and players are grouped into Low, Medium, and High rating classes using the 33% and 66% quantiles.

- SVM-Based Classification and Final Score

  A Support Vector Machine (SVM) with an RBF kernel is trained using overall metrics to classify players into the rating groups (Low/Medium/High). The SVM prediction is converted into a class score (Low=20, Medium=60, High=100). A final player score is computed using the formula:
  70% performance score + 30% SVM predicted class score.

- Visualization

  Graphs are plotted for all the metrics, Top 10 players for each position and Top 10 overall players. The final score for top players is printed.

5.3 Basketball Player Performance Analysis

- Data Cleaning and Normalization

  Load the NBA dataset and standardize the key performance parameters (PTS, AST, TRB, STL, BLK, FG%) by converting inconsistent formats (%, commas, missing values) to numeric values. Normalize the metrics using MinMaxScaler to bring all the values to a standard 0–1 scale.

- Performance Score calculation

  Calculate a performance score (0–100) for each player by normalizing the statistics and averaging them. Group the players into Low, Medium, High rating classes using quantile thresholds (33% & 66%).

- SVM based Calculation

  Train a Support Vector Machine (RBF kernel) using the six performance metrics. Predict the class (Low/Medium/High) for each player using the SVM and convert the predicted class into a score-based weighting system (20/60/100). A final score is calculated using 0.7 × performance score + 0.3 × SVM predicted score.

- Visualization

  Graphs are plotted for all the 6 metrics, Top 10 players for each position and Top 10 overall players. The final score for top players is printed.

6. Implementation

6.1 Football Player performance analysis

The proposed system is implemented entirely in Python, utilizing its robust ecosystem of data science and machine learning libraries.

6.1.1 Software and Libraries Used

- ❖ The analysis was implemented using Python 3.x for data analysis and machine learning.
- ❖ Pandas was used for structured data handling, cleaning, and preprocessing.
- ❖ NumPy supported numerical operations and vectorized computations.
- ❖ Scikit-Learn provided machine learning tools such as MinMaxScaler, StandardScaler, and Support Vector Machine (SVM) classifiers.
- ❖ Matplotlib and Seaborn were employed for graphical visualisation, including distribution plots and confusion matrices.

6.1.2 Dataset and Parameters

- ❖ The dataset consists of FIFA player statistics, including individual performance attributes across attacking, midfield, defensive, and goalkeeping roles.
- ❖ Key parameters selected for modelling include:
  - Goals Scored
  - Assists Provided
  - Dribbles per 90
  - Interceptions per 90
  - Tackles per 90

- Total Duels Won per 90

6.1.3 Implementation Workflow

- Data Cleaning:
  Loaded the dataset and cleaned all performance metrics by converting missing or invalid values into numeric form.
- Normalization & Scoring:
  Normalized selected player parameters and computed a composite performance score.
- SVM Classification:
  Trained an SVM model to classify players into Low, Medium, and High classes and generated final scores.
- Evaluation:
  Assessed model accuracy using classification metrics and a confusion matrix.
- Visualization & Ranking:
  Plotted parameter distributions and confusion heatmaps, then generated final player rankings by position and overall score.

6.1.4 Output

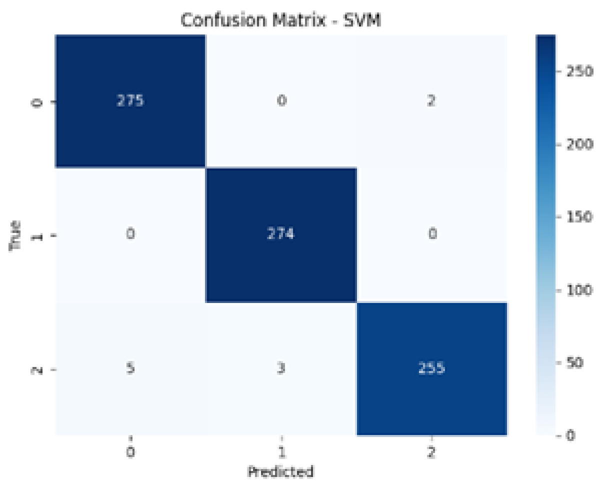


Figure 1: SVM confusion matrix for FIFA Dataset

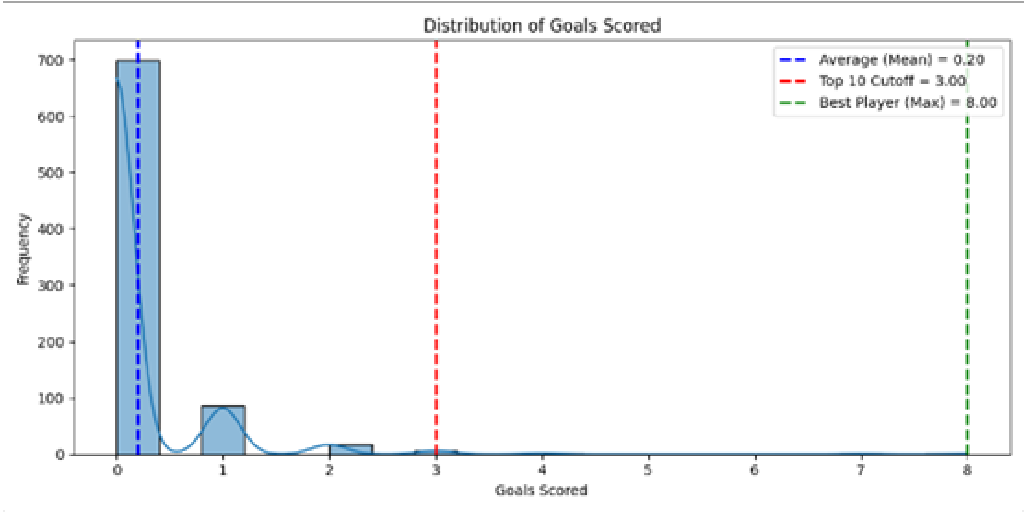


Figure 2: Distribution of goals scored

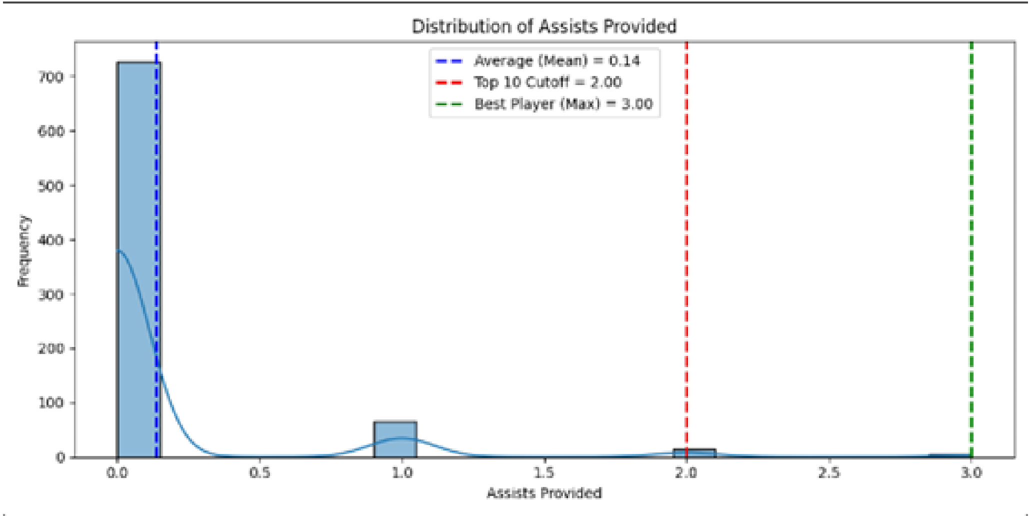


Figure 3: Distribution of Assists provided

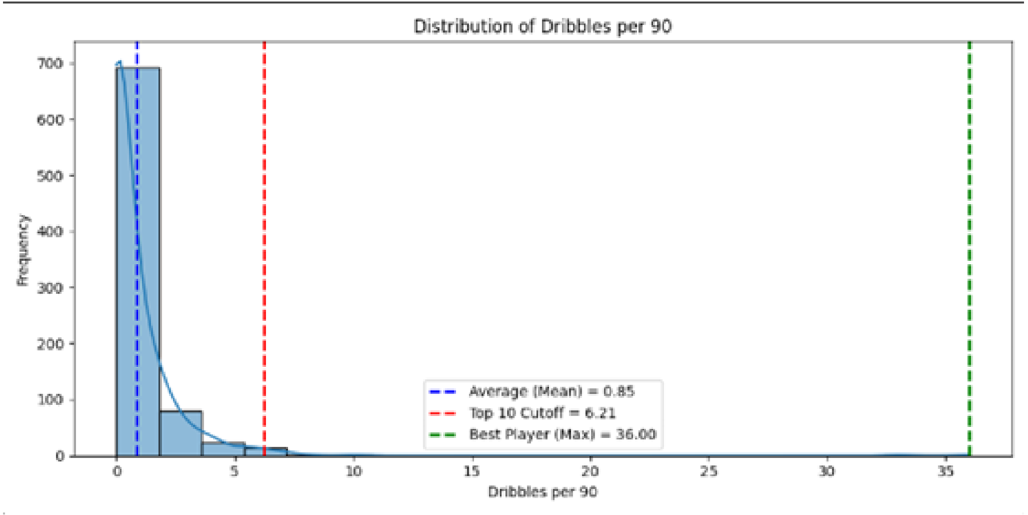


Figure 4: Distribution of dribbles per 90 minutes

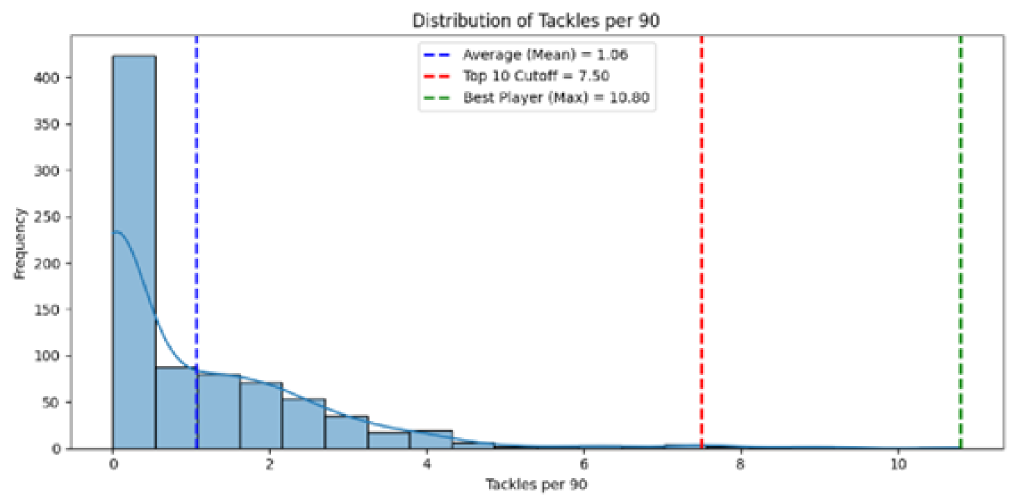


Figure 5: Distribution of tackles per 90 minutes

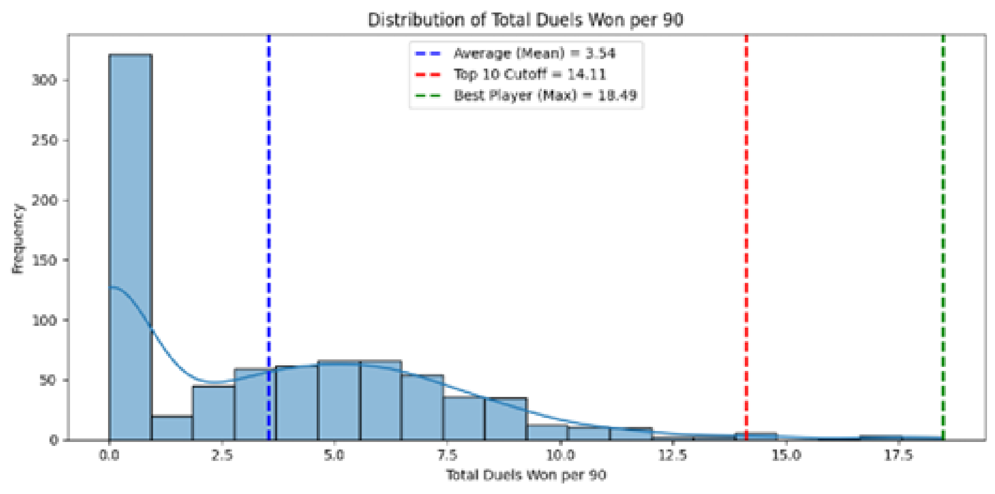


Figure 6: Distribution of duels won per 90 minutes

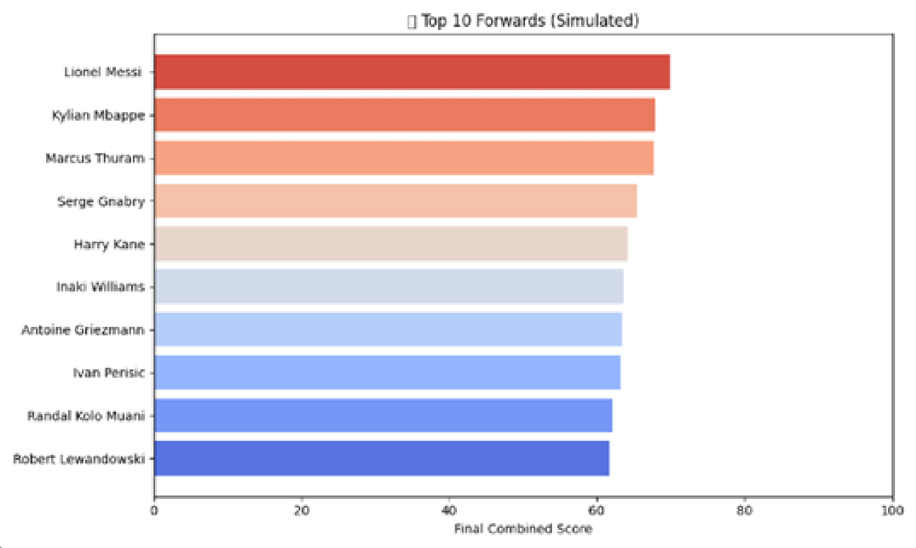


Figure 7: Top 10 forwards

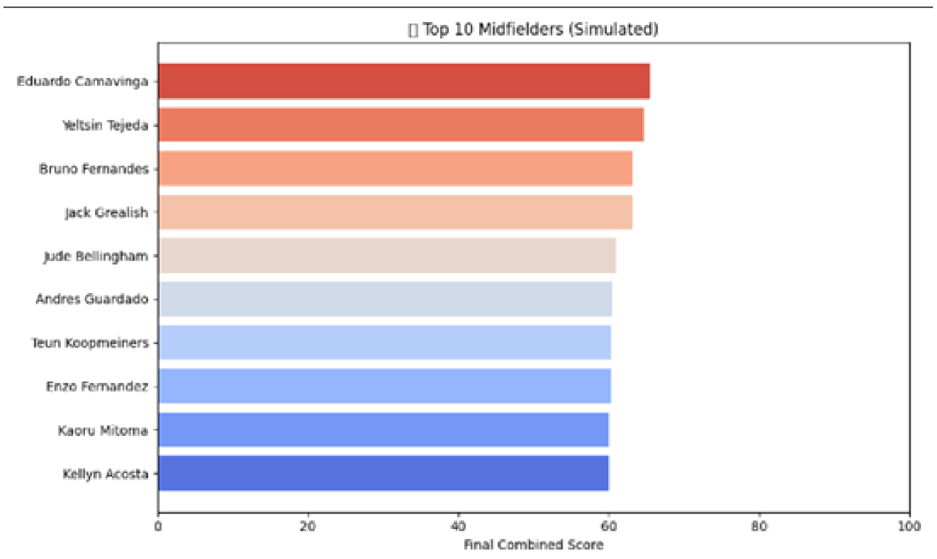


Figure 8: Top 10 midfielders

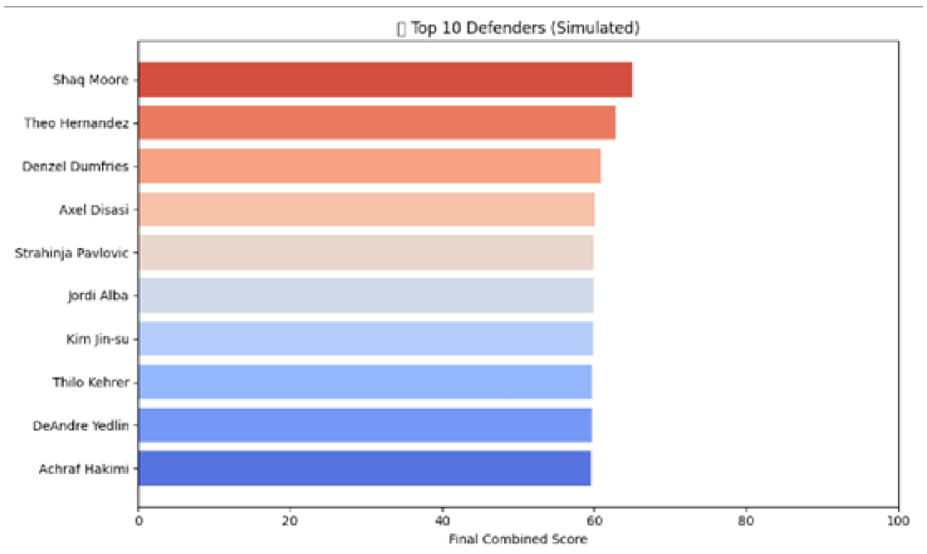


Figure 9: Top 10 defenders

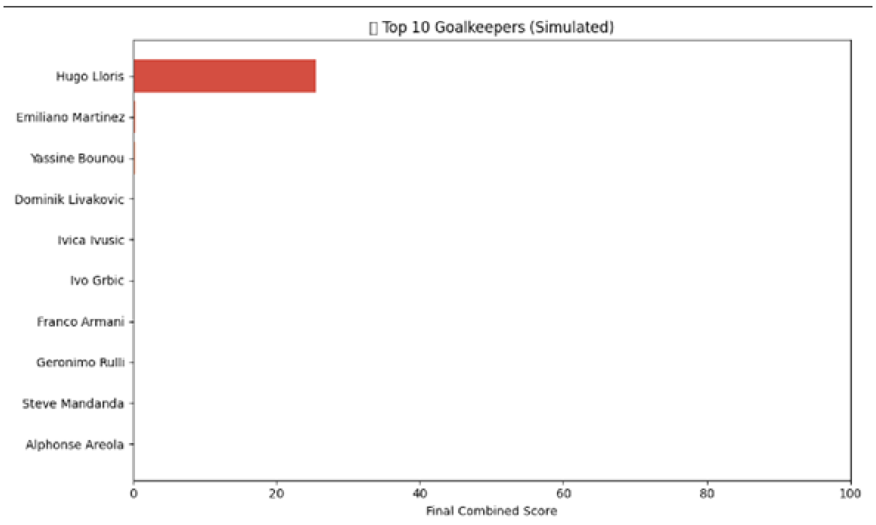


Figure 10: Top 10 goalkeepers

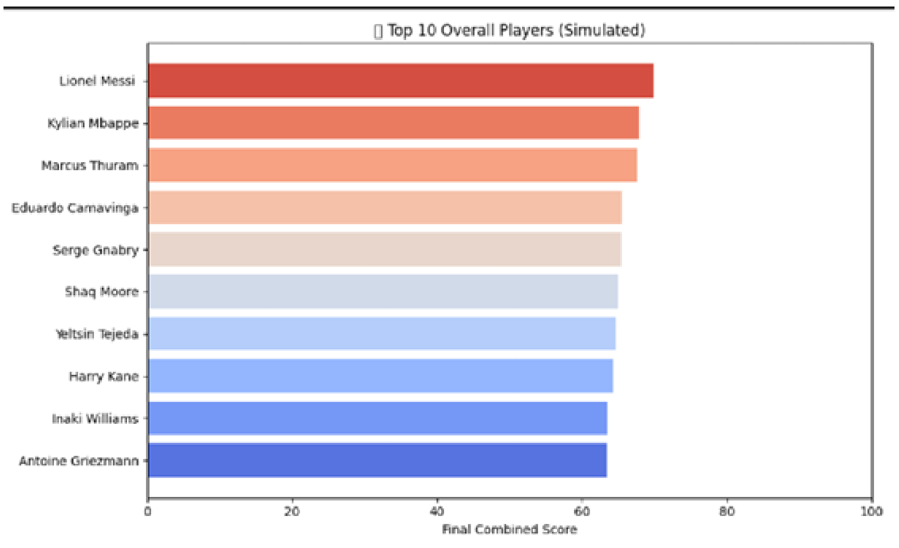


Figure 11: Top 10 overall players

6.1.5 Computational workflow

- Data Loading and Preprocessing:

  Load, clean, and preprocess the FIFA dataset by converting all performance metrics into numerical values while handling missing or invalid entries.

- Metric Normalization and Score Generation:

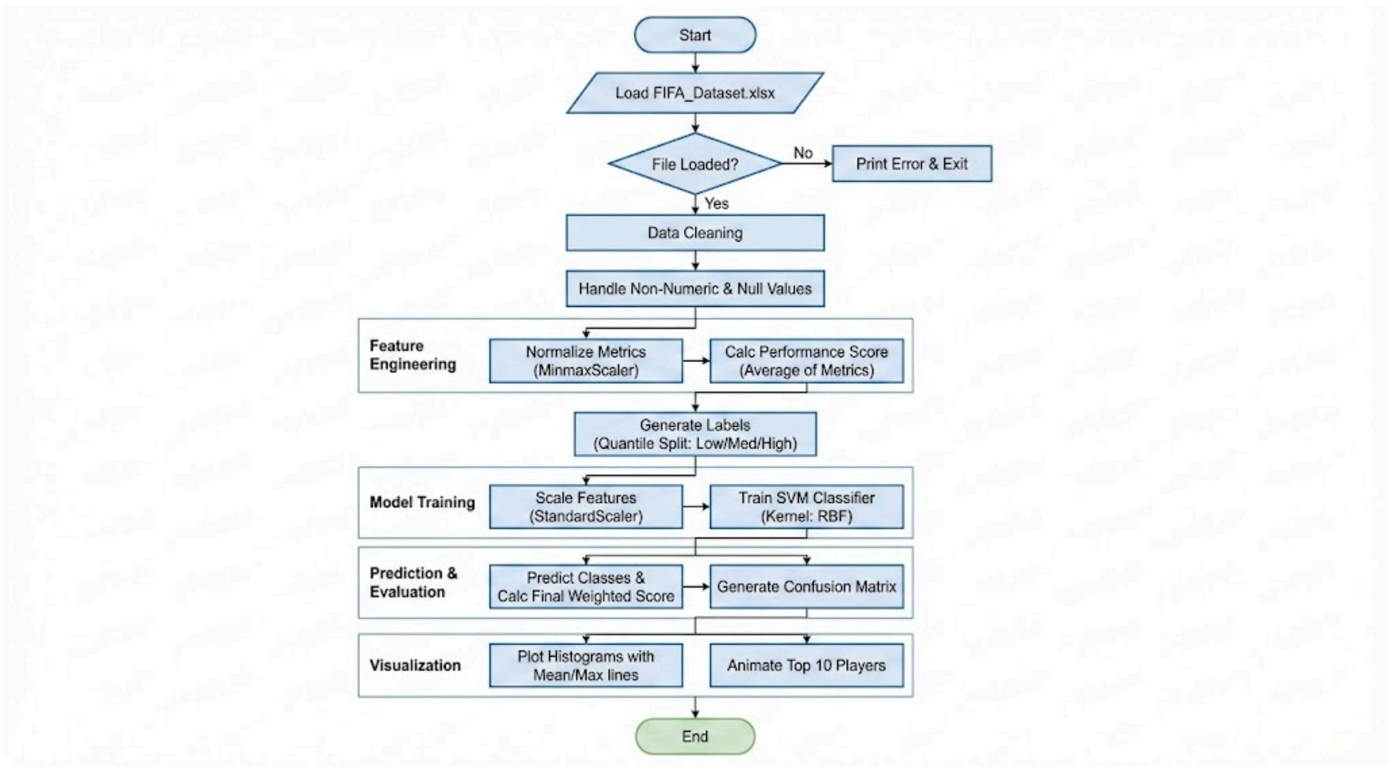
  Normalize selected performance features using MinMax scaling and compute an overall performance score and then classify the player performance categories into Low, Medium, and High classes.

- SVM Training and Prediction:

  Train an SVM model with an RBF kernel on the standardized metrics to classify player performance categories and generate predicted class scores

- Evaluation, Visualization, and Ranking:

  Evaluate model accuracy using classification metrics and confusion matrix visualization, and then rank players based on a combined score from performance metrics and SVM predictions.



Start
Load FIFA_Dataset.xlsx
File Loaded?
No
Print Error & Exit
Yes
Data Cleaning
Handle Non-Numeric & Null Values
Feature Engineering
Normalize Metrics (MinmaxScaler)
Calc Performance Score (Average of Metrics)
Generate Labels (Quantile Split: Low/Med/High)
Model Training
Scale Features (StandardScaler)
Train SVM Classifier (Kernel: RBF)
Prediction & Evaluation
Predict Classes & Calc Final Weighted Score
Generate Confusion Matrix
Visualization
Plot Histograms with Mean/Max lines
Animate Top 10 Players
End

6.2.Cricket Player performance analysis

6.2.1. Software and libraries used

- ❖ Python 3.x for implementation.
- ❖ Pandas & NumPy for data cleaning and preprocessing.
- ❖ Scikit-Learn for normalization and SVM classification.
- ❖ Matplotlib & Seaborn for visualisation and animations.

6.2.2. Dataset and Parameters

- ❖ IPL dataset containing batting, bowling, and wicket-keeping statistics.
- ❖ Batting metrics: Runs Scored, Batting Average, Strike Rate.
- ❖ Bowling metrics: Wickets, Economy Rate, Bowling Strike Rate (inverted for scoring).
- ❖ Keeping metric: Stumpings for wicketkeeper identification.

6.2.3. Implementation

- ❖ Data Cleaning:
  Loaded the IPL dataset and converted all performance metrics into clean numeric values.
- ❖ Normalization & Scoring:
  Normalized batting, bowling, and overall metrics and generated performance scores.

❖ SVM Classification:
 Classified players into Low/Medium/High groups and computed final combined scores.

❖ Visualization & Ranking:
 Created distribution plots, confusion matrices, and Top-10 ranking animations for each role.

6.2.4. Output

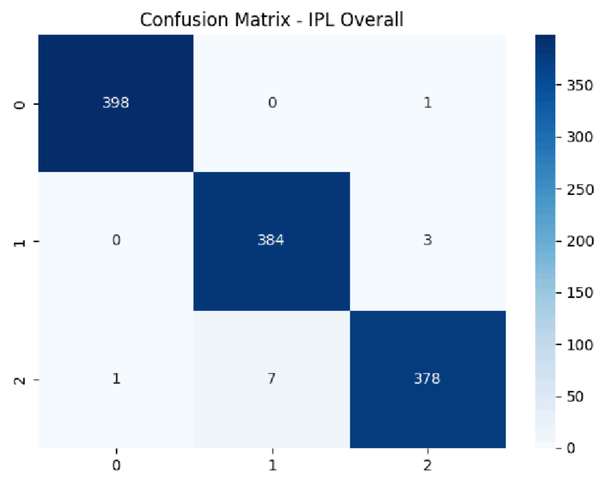


Figure 1: SVM Confusion matrix for IPL dataset

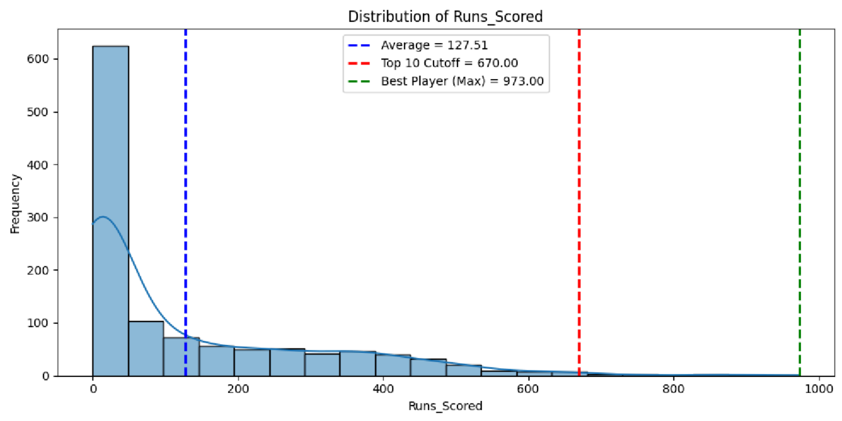


Figure 2: Distribution of runs scored

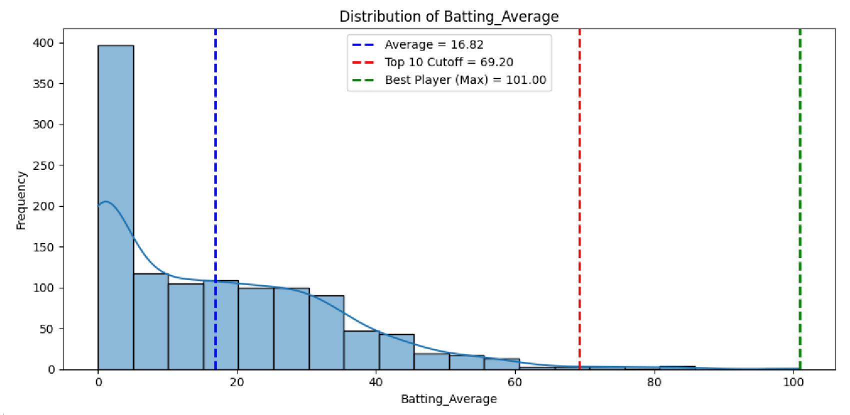


Figure 3: Distribution of batting average

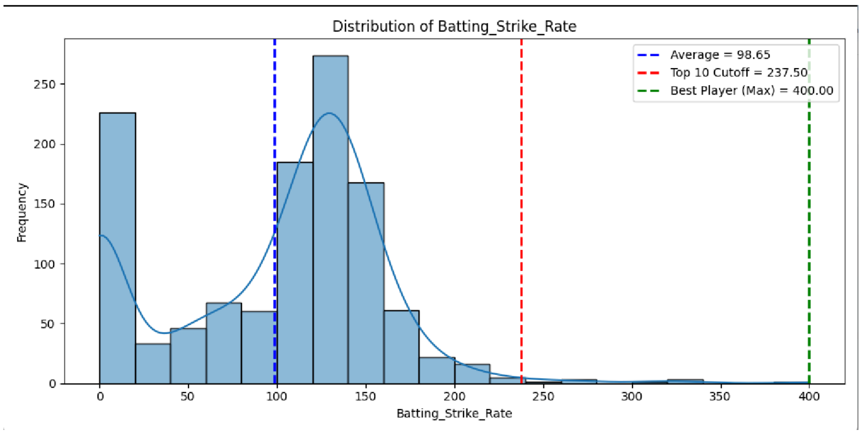


Figure 4: Distribution of bowling strike rate

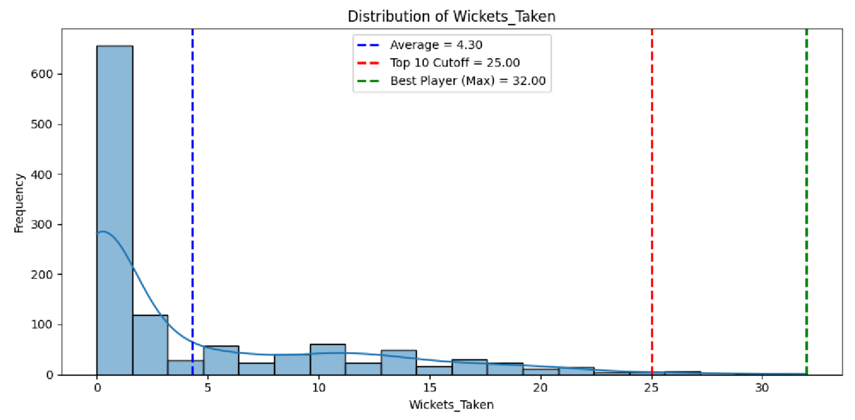


Figure 5: Distribution of Wickets taken

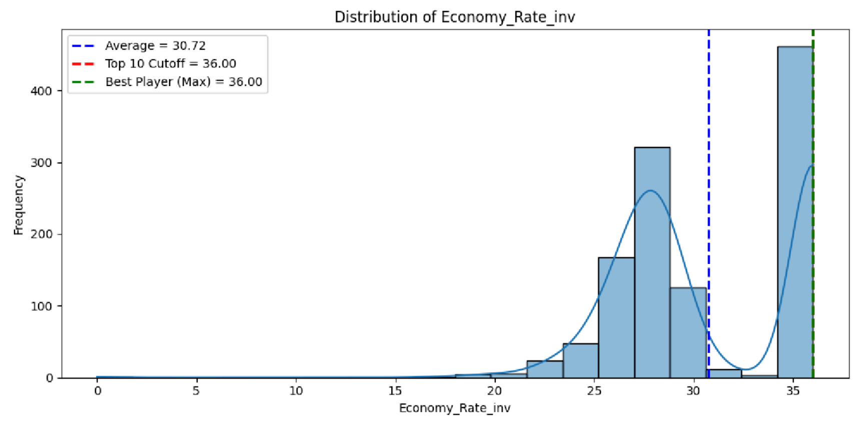


Figure 6: Distribution of economy rate (inversed)

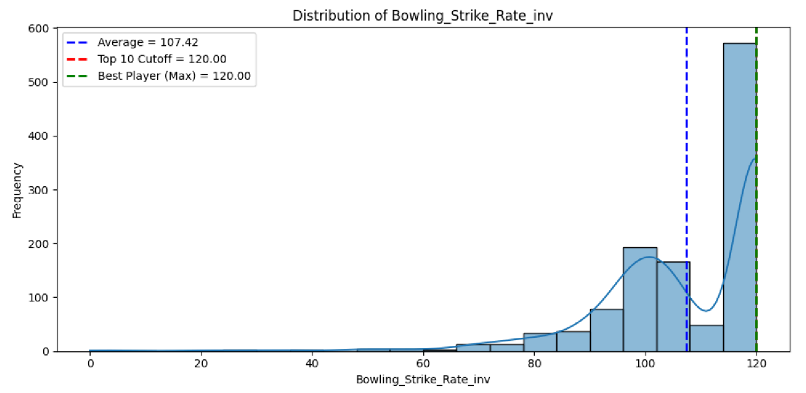


Figure 7: Distribution of bowling strike rate (inversed)

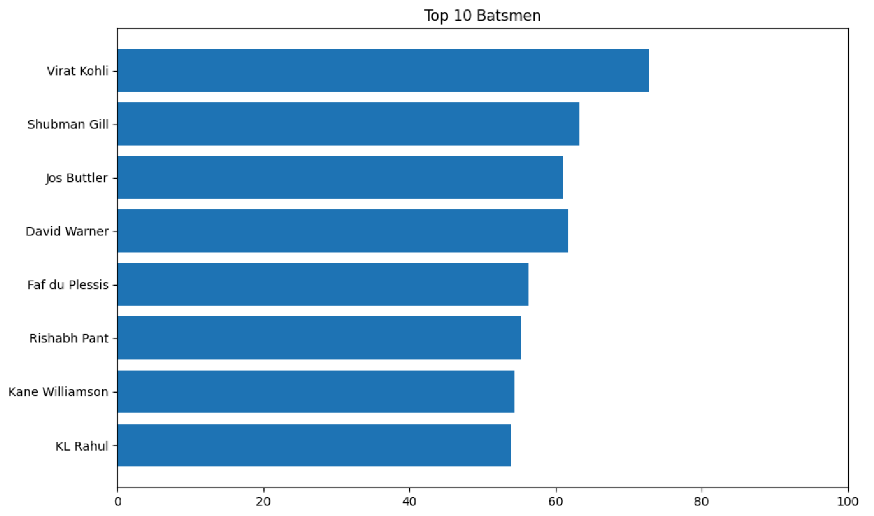


Figure 8: Graph showing top 10 batsmen

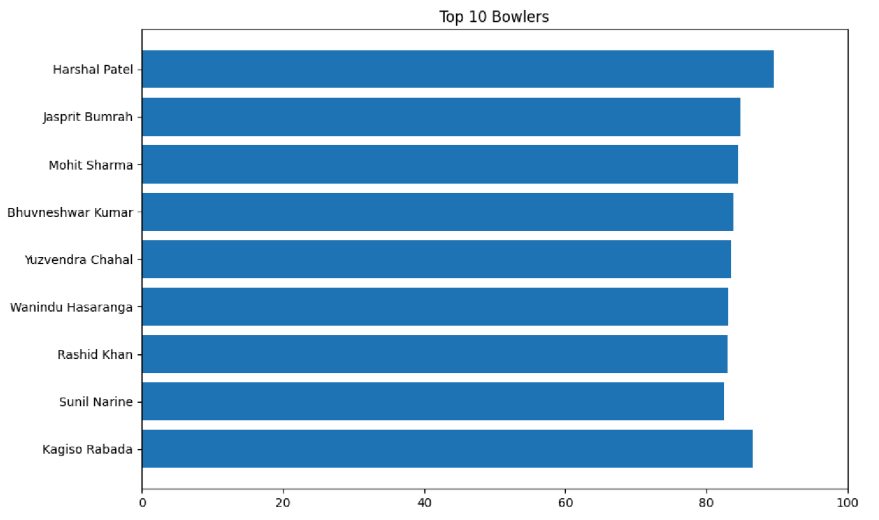


Figure 9: Graph showing top 10 bowlers

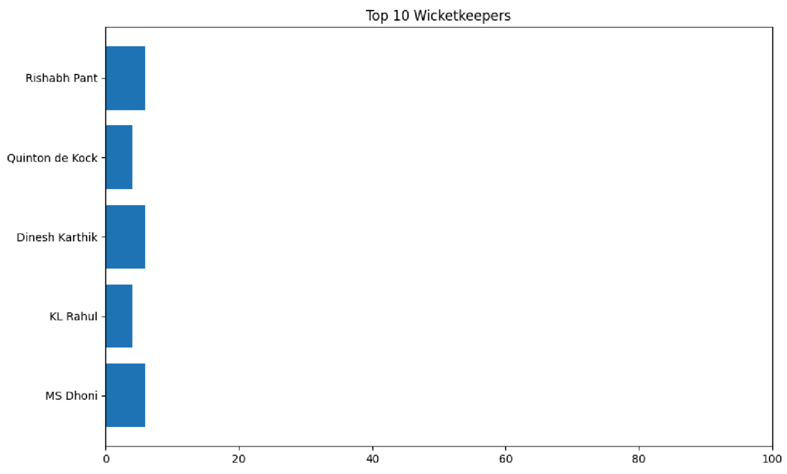


Figure 10: Graph for top 10 wicketkeepers

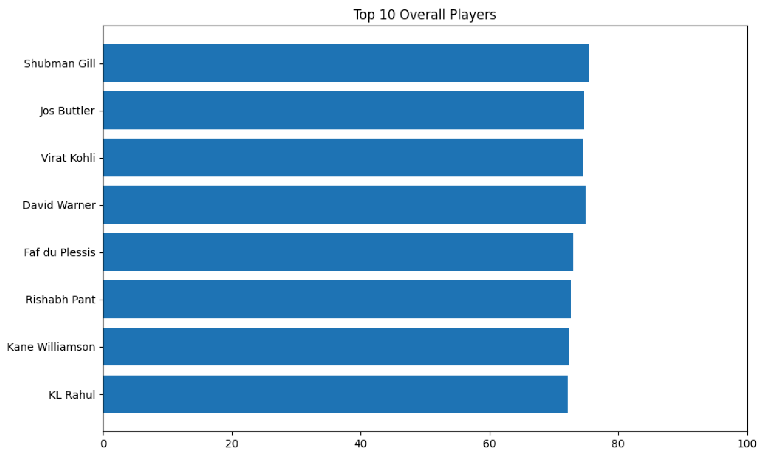


Figure 11: Graph for top 10 overall players

6.2.5 Computational workflow

- Data Loading & Preprocessing

  Load IPL player stats data, clean up inconsistent values, remove commas, handle missing fields, and convert all metrics to numeric.

- Feature Engineering & Normalization

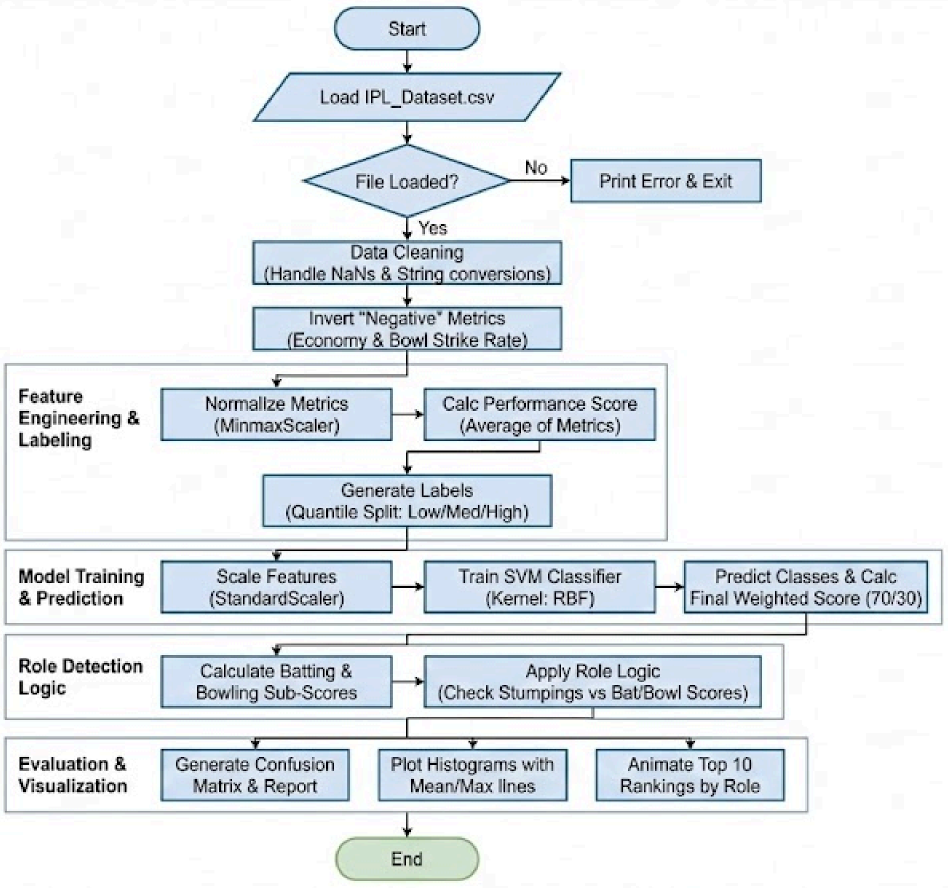
  Create batting, bowling, and keeping performance metrics, normalize features using MinMaxScaler and StandardScaler, and calculate composite performance scores/

- Classification & Scoring

  Classify players into Low/Medium/High rating groups using SVM and combine performance score and predicted class score to generate the final player score.

- Role Detection & Visualization

  Detect player roles (Batsman, Bowler, Wicketkeeper, All-rounder) based on metric dominance and generate distribution plots, confusion matrix, animations, and top-10 rankings



Start
Load IPL_Dataset.csv
File Loaded?
No
Print Error & Exit
Yes
Data Cleaning (Handle NaNs & String conversions)
Invert "Negative" Metrics (Economy & Bowl Strike Rate)
Feature Engineering & Labeling
Normalize Metrics (MinmaxScaler)
Calc Performance Score (Average of Metrics)
Generate Labels (Quantile Split: Low/Med/High)
Model Training & Prediction
Scale Features (StandardScaler)
Train SVM Classifier (Kernel: RBF)
Predict Classes & Calc Final Weighted Score (70/30)
Role Detection Logic
Calculate Batting & Bowling Sub-Scores
Apply Role Logic (Check Stumpings vs Bat/Bowl Scores)
Evaluation & Visualization
Generate Confusion Matrix & Report
Plot Histograms with Mean/Max lines
Animate Top 10 Rankings by Role
End

6.3 Basketball player performance analysis

6.3.1. Software and Libraries used

- Python 3.x for implementation.
- Pandas & NumPy for data cleaning and preprocessing.
- Scikit-Learn for normalization, SVM classification, and model evaluation.
- Matplotlib & Seaborn for visualisation of distributions, confusion matrices, and animations.
- Matplotlib Animation for generating Top-10 dynamic ranking charts.

6.3.2. Dataset and parameters

- NBA dataset containing player performance statistics and positional data.
- Six key performance parameters used: Points (PTS), Assists (AST), Rebounds (TRB), Steals (STL), Blocks (BLK), Field-Goal Percentage (FG%).
- Additional fields such as Player Name and Position (PG, SG, SF, PF, C) were used for grouping into Guard, Forward, and Center.
- Percentage and string-based metrics were cleaned and converted into numerical form.

6.3.3. Implementation

- Data Cleaning:
  Converted all performance metrics into numeric values and standardized FG% for consistency.
- Normalization & Scoring:
  Normalized six performance parameters and computed a composite player performance score (0–100).
- SVM Classification:
  Classified players into Low/Medium/High categories using an SVM classifier and generated a blended final score.
- Visualization & Ranking:
  Produced distribution graphs, confusion matrices, and animated Top-10 rankings for Guards, Forwards, Centers, and Overall players.

6.3.4.Output

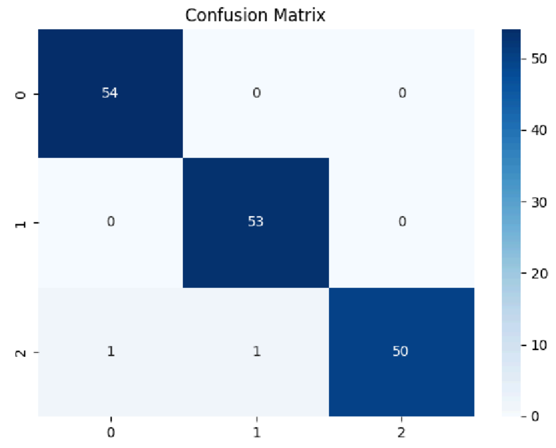


Figure 1: SVM confusion matrix for NBA dataset

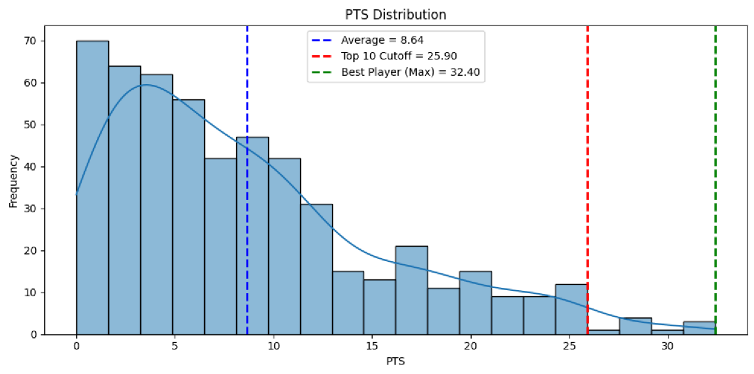


Figure 2: Points distribution

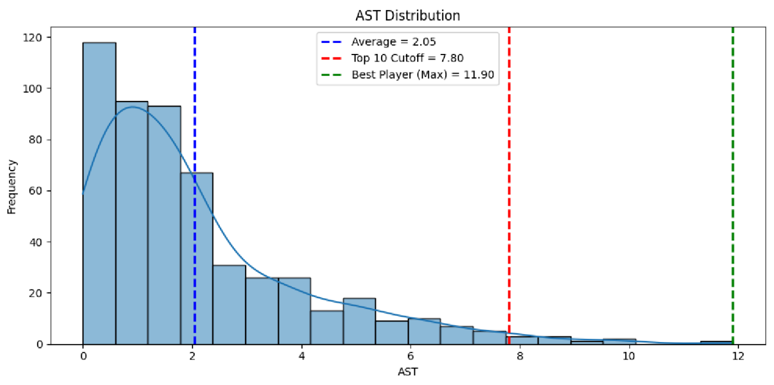


Figure 3: Assists distribution

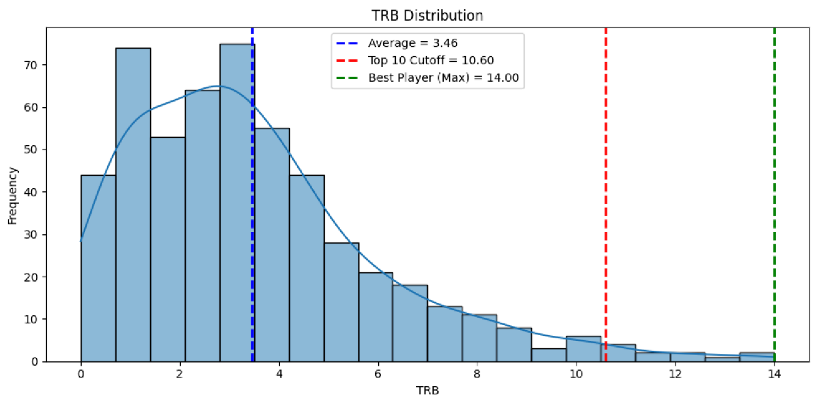


Figure 4: Rebounds distribution

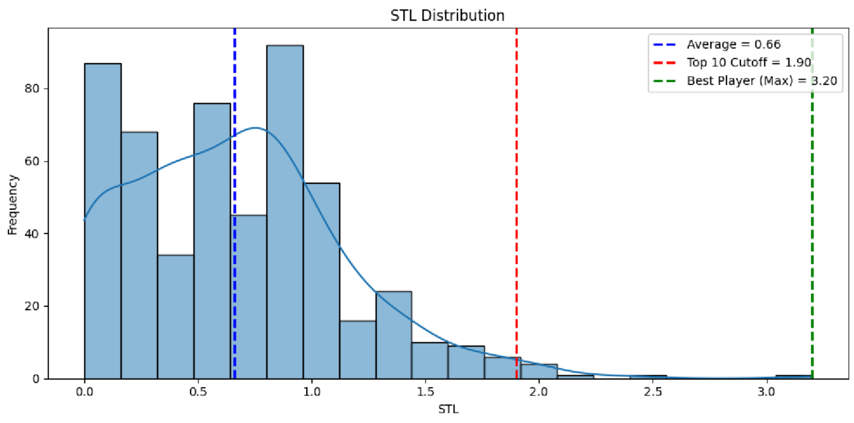


Figure 5: Steals distribution

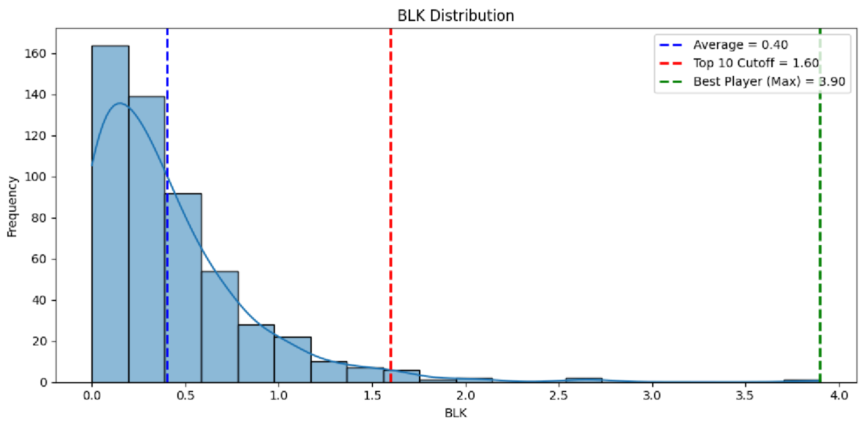


Figure 6: Blocks distribution

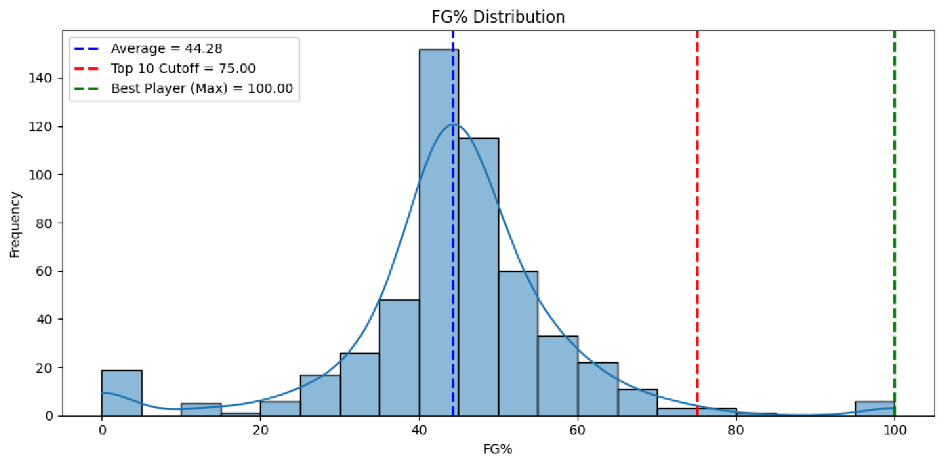


Figure 7: Field Goal% Distribution

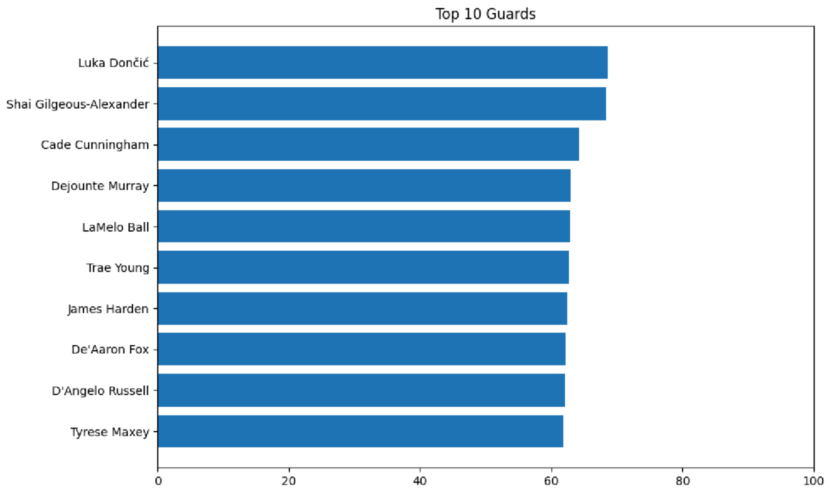


Figure 8: Top 10 Guards

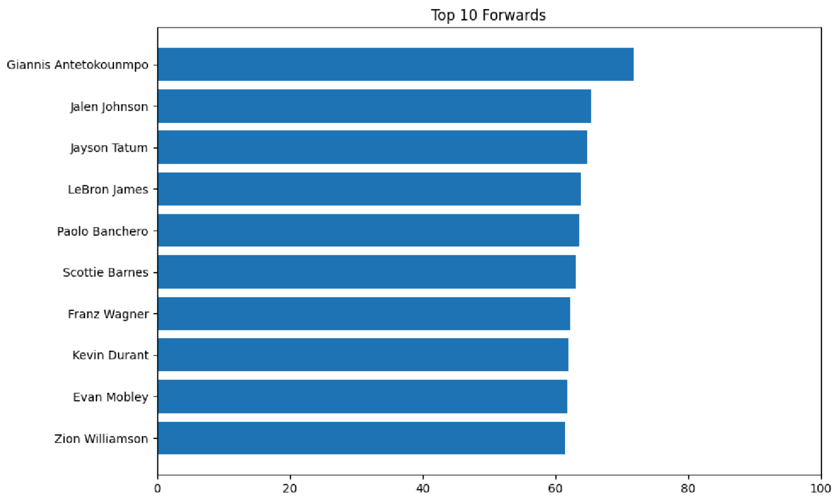


Figure 9: Top 10 forwards

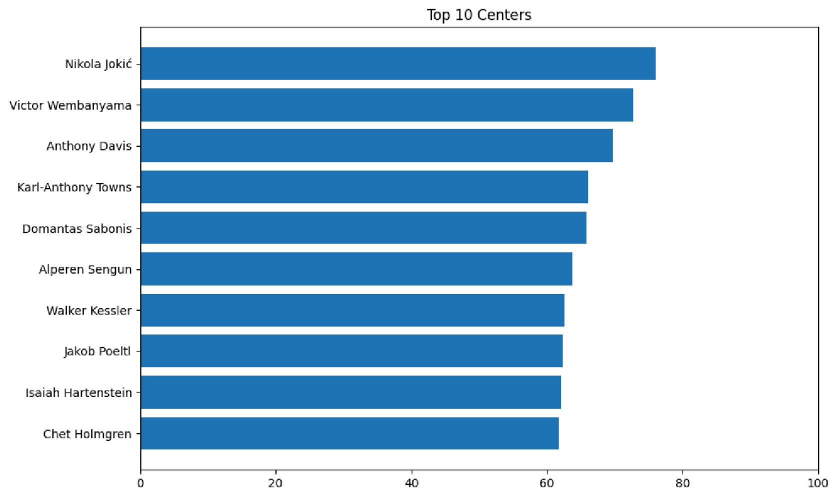


Figure 10: Top 10 centers

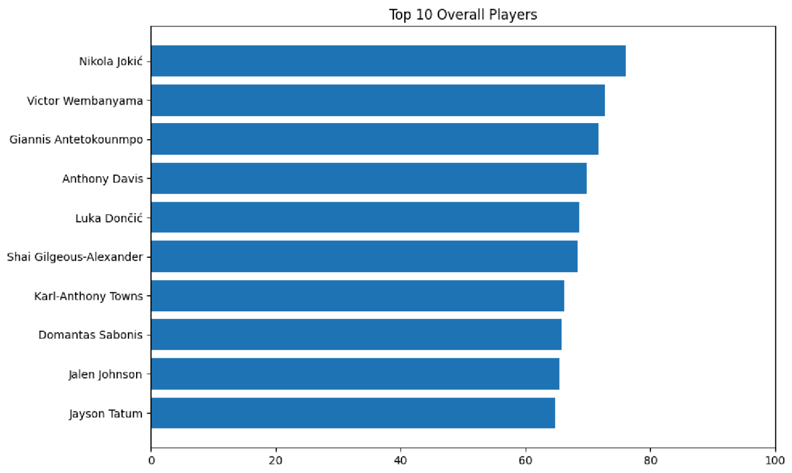


Figure 11: Top 10 overall players

6.3.5 Computational workflow

- Data Cleaning and Feature Preparation:

  Load the dataset, clean it (remove invalid or non-numeric values), extract and standardize key NBA performance metrics (PTS, AST, TRB, STL, BLK, FG%).

- Normalization and Score Generation:

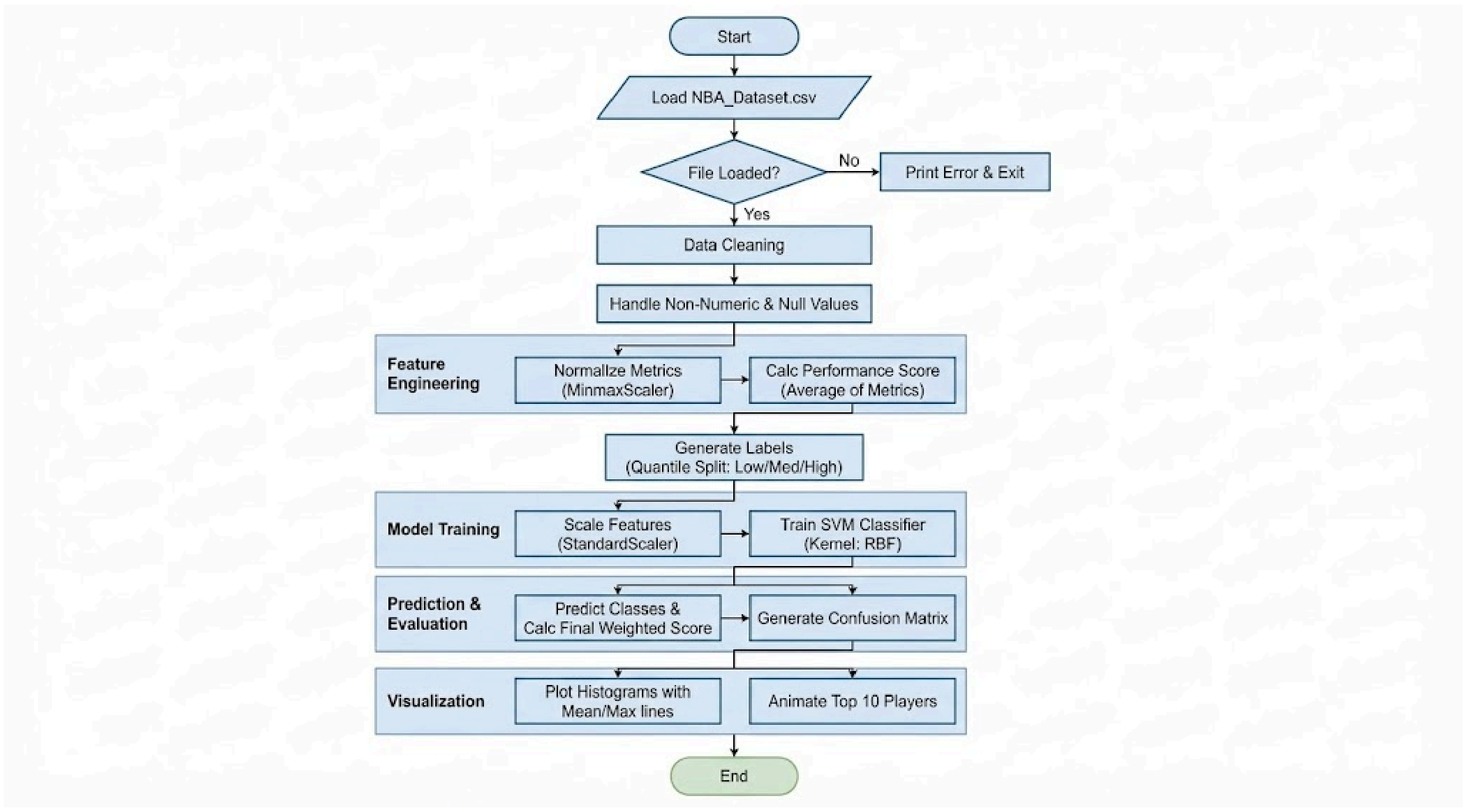
  MinMax scale all metrics and compute a performance score, group players into Low, Medium, and High categories using quantile-based thresholds.

- SVM Model Training and Evaluation:

  Split the dataset into training and testing sets, scale the data using StandardScaler, train an SVM (RBF) model to classify players, generate model accuracy and confusion matrices.

- Final Scoring and Player Ranking:

Convert predicted classes to numeric scores and combine with performance scores to calculate a final ranking score; visualize top players (overall and position-wise) with bar animations and statistical distributions.


Start
Load NBA_Dataset.csv
File Loaded?
No
Print Error & Exit
Yes
Data Cleaning
Handle Non-Numeric & Null Values
Feature Engineering
Normalize Metrics (MinmaxScaler)
Calc Performance Score (Average of Metrics)
Generate Labels (Quantile Split: Low/Med/High)
Model Training
Scale Features (StandardScaler)
Train SVM Classifier (Kernel: RBF)
Prediction & Evaluation
Predict Classes & Calc Final Weighted Score
Generate Confusion Matrix
Visualization
Plot Histograms with Mean/Max lines
Animate Top 10 Players
End

7. Results and Analysis

The proposed performance evaluation framework was applied to datasets from three major sports (cricket, football, and basketball) using the same modelling pipeline of normalization, quantile-based classification, and SVM prediction. The combined statistical–machine learning scoring system produced highly accurate and stable player classifications in all domains.

7.1 Football Player performance analysis

- The football model classified the players with 98.77% accuracy, indicating the selected metrics represent player performance very well.
- The chosen 6 metrics helped the model classify players fairly
- Players were grouped into low, medium and high classes
- The combination of performance score and SVM prediction provided more precise rankings than using raw statistics.

7.2 Cricket Player performance analysis

- The model classified players into Low, Medium, and High performers with 98.98% accuracy.
- Batting, bowling, and wicketkeeping metrics worked well together to separate different types of players.
- Role-based scoring (Batsman, Bowler, All-Rounder, Wicketkeeper) produced rankings that closely followed real on-field performances.
- The final Top-10 lists identified the best players in each category.

.

7.3 Basketball Player performance analysis

- The NBA model classified 98.74% of the players correctly, demonstrating that the method also works with mixed-type stats such as PTS, AST, STL, BLK, and FG%.
- The model separated top performers from average players by grouping players into Guards, Forwards, and Centers
- The distribution graphs distinguished top performers from the rest.

8. Error Analysis

8.1 Football player performance analysis

- The IPL model achieved 98.98% accuracy, meaning very few players were misclassified.
- The confusion matrix showed only a small number of cases where Medium performers were classified as High or Low.
- Most errors happened with all-rounders, since they have mixed batting and bowling strengths, making them harder for the model to categorize.
- Overall, the low error rate shows that the chosen metrics (runs, wickets, economy, strike rate, stumpings) represent player performance very well.

8.2.Cricket player performance analysis

- The football model achieved 98.77% accuracy, with only slight misclassification between Medium and High performers.
- Errors mostly occurred for players with balanced stats across categories (e.g., moderate goals + moderate defensive stats), making their class boundary unclear.
- The confusion matrix showed almost no misclassification for Low performers, meaning the model is very good at identifying weaker players.
- The low error rate confirms that the six football metrics gave a clear separation between performance levels.
- The model identifies wicketkeepers only when they have at least one recorded stumping, so players who are real wicketkeepers but have zero stumpings in the dataset are misclassified - resulting in few wicketkeepers being detected.

8.3 Basketball Player Performance analysis

- The NBA model reached 98.74% accuracy, with minimal confusion between Medium and High categories.
- Players who excel only in one area (e.g., high blocks but low assists) cause most misclassifications because their profiles differ from balanced players.
- Guards had slightly higher classification errors due to high variability in stats like assists and steals.
- Overall, the model showed strong consistency even with mixed metrics like FG%, rebounds, and steals.

9. Conclusion

In this study, a comprehensive machine-learning framework for player performance evaluation across cricket, football, and basketball was introduced, which produced reliable performance scores and accurate player groupings through data cleaning, normalization, and SVM-based classification, and demonstrated high accuracy (98.98% in cricket, 98.77% in football, and 98.74% in basketball), which confirmed the robustness of the approach across different sports datasets and helped interpretability through visual tools, such as distribution plots, confusion matrices, and ranking animations. In summary, this study presents a scalable and sport-independent method for objective player performance evaluation and comparative analysis.

10. Future Work Implementation

- **Add Age Metrics:** Use player age, experience, and career stage in the scoring model, develop age-adjusted performance scores, and peak-age curves to achieve fairness and improve prediction accuracy.

- **Dynamically Calculate Performance on a Match-by-Match Basis:** Change from season-level aggregates to last-N-match averages, momentum indicators, and time-series tracking.

- **Enhanced Feature Engineering:** Introduce advanced metrics like per-possession efficiency, defensive impact ratings, or situational performance to increase model depth.

- **Model Optimization and Comparative Analysis:** Conduct hyperparameter tuning and compare SVM with other models (Random Forest, Gradient Boosting, and Neural Networks) to determine the most effective classifier.

- **Interactive Dashboard and Real-Time Visualization:** Create a dashboard to monitor player performance, visualize trends, rankings, and updated scores after each match.

11. Acknowledgement

The authors would like to express their earnest gratitude to the School of Computing and the faculty of Amrita Vishwa Vidyapeetham for their guidance and support throughout this project. We are peculiarly appreciative to Ms. Mamatha T. M., our course mentor, for her continuous mentorship, theoretical guidance, and encouragement, which were priceless in transforming our initial idea into a functional system. We also appreciate the theoretical environment and research oriented culture provided by the Department of CSE/AI, which motivated us to rigorously explore the methodology, frame research problems, and validate our approach mathematically. The resources, insights, and encouragement provided by the department and faculty played a pivotal role in the productive completion of this study.

12. Contributions

Surya Pramod led the analysis of the FIFA and NBA datasets, where he implemented the performance-scoring algorithms and conducted the error analysis for the machine-learning model. He also contributed significantly to formulating the future work scope and refining the overall evaluation pipeline for cross-sport performance assessment.

Vijaynath handled the IPL cricket dataset and performed the literature review, identifying research gaps in existing sports-analytics methodologies. He further contributed to the design of the normalization algorithm used for metric scaling and supported the methodological development by aligning the framework with established research practices.

13. Appendix

12.1 Football Dataset

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import matplotlib.animation as animation

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score


#Load Data

file_path = "FIFA_Dataset.xlsx"

df = pd.read_excel(file_path)

df.columns = [c.strip() for c in df.columns]


metric_cols = [
```

```python
    'Goals Scored', 'Assists Provided', 'Dribbles per 90',
    'Interceptions per 90', 'Tackles per 90', 'Total Duels Won per 90'
]

def clean_numeric(x):
    if pd.isna(x): return 0.0
    if isinstance(x, str):
        x = x.strip()
        if x in ['-', 'N.A', 'NA', 'nan']:
            return 0.0
        try:
            return float(x.replace(',', ''))
        except:
            return 0.0
    return float(x)

for col in metric_cols:
    if col not in df.columns:
        df[col] = 0.0
    df[col] = df[col].apply(clean_numeric)

df[metric_cols] = df[metric_cols].fillna(0)

scaler_norm = MinMaxScaler()
df_norm     =     pd.DataFrame(scaler_norm.fit_transform(df[metric_cols]),
columns=metric_cols)

df['performance_score'] = df_norm.mean(axis=1) * 100

# Labeling based on performance score
q33, q66 = df['performance_score'].quantile([0.33, 0.66])

df['rating_class'] = df['performance_score'].apply(
   lambda x: 'Low' if x <= q33 else ('Medium' if x <= q66 else 'High')

)


X = df[metric_cols]

y = df['rating_class']


scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


svm_clf = SVC(kernel='rbf', C=10, gamma='scale')

svm_clf.fit(X_scaled, y)


df['predicted_class'] = svm_clf.predict(X_scaled)


df['predicted_score'] = df['predicted_class'].map({

   'Low': 0,

   'Medium': 50,

   'High': 100

})


df['final_score'] = 0.5 * df['performance_score'] + 0.5 *
df['predicted_score']


print("\nSVM CLASSIFICATION REPORT:")

print(classification_report(y, df['predicted_class']))


cm = confusion_matrix(y, df['predicted_class'])

acc = accuracy_score(y, df['predicted_class'])


print("\nCONFUSION MATRIX:")

print(cm)

print(f"\nACCURACY: {acc*100:.2f}%")

print(f"ERROR RATE: {(1-acc)*100:.2f}%")
```

```python
# Plot Confusion Matrix
plt.figure(figsize=(7,5))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.title("Confusion Matrix - SVM")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

def plot_parameter_distributions():
   for col in metric_cols:
       plt.figure(figsize=(10, 5))
       sns.histplot(df[col], kde=True, bins=20)

       # BLUE LINE = Mean
       avg = df[col].mean()

       # GREEN LINE = Max
       max_val = df[col].max()

       # RED LINE = 10th highest value
       sorted_vals = df[col].sort_values(ascending=False).values
       if len(sorted_vals) >= 10:
           top10_cutoff = sorted_vals[9]
       else:
           top10_cutoff = sorted_vals[-1]

       plt.axvline(avg, color='blue', linestyle='--', linewidth=2,
                   label=f"Average (Mean) = {avg:.2f}")

                plt.axvline(top10_cutoff,   color='red',   linestyle='--',
linewidth=2,
                   label=f"Top 10 Cutoff = {top10_cutoff:.2f}")
```

```
        plt.axvline(max_val, color='green', linestyle='--', linewidth=2,
                    label=f"Best Player (Max) = {max_val:.2f}")

        plt.title(f"Distribution of {col}")
        plt.xlabel(col)
        plt.ylabel("Frequency")
        plt.legend()
        plt.tight_layout()
        plt.show()

print("\nGenerating parameter distribution graphs...")
plot_parameter_distributions()

def animate_top10(position_code, title_name):
    if position_code == 'All':
        top10 = df.sort_values('final_score', ascending=False).head(10)
    else:
        top10 = df[df['Position'] ==
position_code].sort_values('final_score', ascending=False).head(10)

    if top10.empty:
        print(f"No players found for {title_name}")
        return

    top10 = top10.sort_values('final_score')

    fig, ax = plt.subplots(figsize=(10, 6))
    bars = ax.barh(top10['Player Name'], [0]*len(top10),
                  color=sns.color_palette("coolwarm", len(top10)))

    ax.set_xlim(0, 100)
    ax.set_xlabel("Final Combined Score")
```

```python
   ax.set_title(f"Top 10 {title_name} (Simulated)")


   def update(frame):

       for bar, score in zip(bars, top10['final_score']):

           bar.set_width(score * (frame / 100))

       return bars


   ani = animation.FuncAnimation(

       fig, update, frames=np.linspace(0, 100, 60),

       interval=25, blit=False, repeat=False

   )


   plt.tight_layout()

   plt.show()


animate_top10('FW', 'Forwards')

animate_top10('MF', 'Midfielders')

animate_top10('DF', 'Defenders')

animate_top10('GK', 'Goalkeepers')

animate_top10('All', 'Overall Players')


print("TOP 10 FORWARDS (FW)")

print(df[df['Position']=='FW'].sort_values('final_score',
ascending=False).head(10)[[

   'Player Name', 'final_score', 'performance_score', 'predicted_class'

]])


print("TOP 10 MIDFIELDERS (MF)")

print(df[df['Position']=='MF'].sort_values('final_score',
ascending=False).head(10)[[

   'Player Name', 'final_score', 'performance_score', 'predicted_class'

]])
```

```
print("TOP 10 DEFENDERS (DF)")

print(df[df['Position']=='DF'].sort_values('final_score',
ascending=False).head(10)[[

    'Player Name', 'final_score', 'performance_score', 'predicted_class'

]])


print("TOP 10 GOALKEEPERS (GK)")

print(df[df['Position']=='GK'].sort_values('final_score',
ascending=False).head(10)[[

    'Player Name', 'final_score', 'performance_score', 'predicted_class'

]])


print("TOP 10 OVERALL PLAYERS")

print(df.sort_values('final_score', ascending=False).head(10)[[

        'Player  Name',  'Position',  'final_score',  'performance_score',
'predicted_class'

]])
```

12.2 Cricket Dataset

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import matplotlib.animation as animation

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.svm import SVC

from  sklearn.metrics  import  classification_report,  confusion_matrix,
accuracy_score


#Load Data

file_path = "IPL_Dataset.csv"

df = pd.read_csv(file_path)

df.columns = [c.strip() for c in df.columns]
```

```python
batting_cols = ['Runs_Scored', 'Batting_Average', 'Batting_Strike_Rate']

bowling_cols = ['Wickets_Taken', 'Economy_Rate', 'Bowling_Strike_Rate']

keeping_cols = ['Stumpings']


all_metrics = batting_cols + bowling_cols + keeping_cols



def clean(x):

   if pd.isna(x): return 0

   if isinstance(x, str):

       x = x.replace(",", "").strip()

       try:

           return float(x)

       except:

           return 0

   return float(x)



for col in all_metrics:

   df[col] = df[col].apply(clean)


df['Economy_Rate_inv'] = df['Economy_Rate'].max() - df['Economy_Rate']

df['Bowling_Strike_Rate_inv']    =    df['Bowling_Strike_Rate'].max()    -
df['Bowling_Strike_Rate']


overall_metric_cols = [

   'Runs_Scored', 'Batting_Average', 'Batting_Strike_Rate',

   'Wickets_Taken', 'Economy_Rate_inv', 'Bowling_Strike_Rate_inv'

]


norm = MinMaxScaler()

df_norm    =    pd.DataFrame(norm.fit_transform(df[overall_metric_cols]),
columns=overall_metric_cols)
```

```
df['performance_score'] = df_norm.mean(axis=1) * 100


q1, q2 = df['performance_score'].quantile([0.33, 0.66])

df['rating_class'] = df['performance_score'].apply(

   lambda x: 'Low' if x <= q1 else ('Medium' if x <= q2 else 'High')

)


X = df[overall_metric_cols]

y = df['rating_class']


sc = StandardScaler()

X_scaled = sc.fit_transform(X)


svm = SVC(kernel='rbf', C=10, gamma='scale')

svm.fit(X_scaled, y)


df['predicted_class'] = svm.predict(X_scaled)

df['predicted_score']  =  df['predicted_class'].map({'Low':  20,  'Medium':
60, 'High': 100})


df['final_score']    =    0.7    *    df['performance_score']    +    0.3    *
df['predicted_score']


# Batting score

bat_norm = MinMaxScaler()

df_bat       =       pd.DataFrame(bat_norm.fit_transform(df[batting_cols]),
columns=batting_cols)

df['batting_score'] = df_bat.mean(axis=1) * 100


# Bowling score

bowl_norm = MinMaxScaler()

df_bowl     =     pd.DataFrame(bowl_norm.fit_transform(df[['Wickets_Taken',
'Economy_Rate_inv', 'Bowling_Strike_Rate_inv']]),

                             columns=['Wickets_Taken', 'Economy_Rate_inv',
'Bowling_Strike_Rate_inv'])
```

```python
df['bowling_score'] = df_bowl.mean(axis=1) * 100



df['keeping_score'] = df['Stumpings']


def detect_role(row):
    if row['Stumpings'] > 0:
        return "Wicketkeeper"

    if row['batting_score'] > row['bowling_score']:
        return "Batsman"

    if row['bowling_score'] > row['batting_score']:
        return "Bowler"

    return "All-Rounder"



df['Role'] = df.apply(detect_role, axis=1)


df['Player_Name'] = df['Player_Name']


print("\nSVM CLASSIFICATION REPORT:")
print(classification_report(y, df['predicted_class']))


cm = confusion_matrix(y, df['predicted_class'])
print("\nCONFUSION MATRIX:")
print(cm)
print(f"\nACCURACY: {accuracy_score(y, df['predicted_class']) * 100:.2f}%")


plt.figure(figsize=(7, 5))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
```

```
plt.title("Confusion Matrix - IPL Overall")

plt.show()


def plot_distributions():

   for col in overall_metric_cols:

       plt.figure(figsize=(10, 5))

       sns.histplot(df[col], kde=True, bins=20)


       avg = df[col].mean()

       max_val = df[col].max()


       sorted_vals = df[col].sort_values(ascending=False).values

           top10_cutoff  =  sorted_vals[9]  if  len(sorted_vals)  >=  10  else
sorted_vals[-1]


             plt.axvline(avg,  color='blue',  linestyle='--',  linewidth=2,
label=f"Average = {avg:.2f}")

                 plt.axvline(top10_cutoff,   color='red',   linestyle='--',
linewidth=2, label=f"Top 10 Cutoff = {top10_cutoff:.2f}")

         plt.axvline(max_val, color='green', linestyle='--', linewidth=2,
label=f"Best Player (Max) = {max_val:.2f}")


       plt.title(f"Distribution of {col}")

       plt.xlabel(col)

       plt.ylabel("Frequency")

       plt.legend()

       plt.tight_layout()

       plt.show()



print("\nGenerating metric distribution graphs...")

plot_distributions()


def animate_top10(scores_col, title):
   top10 = df.sort_values(scores_col, ascending=False).head(10)


   top10 = top10.sort_values(scores_col)


   fig, ax = plt.subplots(figsize=(10, 6))

   bars = ax.barh(top10['Player_Name'], [0] * len(top10))


   ax.set_xlim(0, 100)

   ax.set_title(f"Top 10 {title}")


   def update(frame):

       for bar, score in zip(bars, top10[scores_col]):

           bar.set_width(score * (frame / 100))

       return bars


   ani = animation.FuncAnimation(

             fig, update, frames=np.linspace(0, 100, 60), interval=25,
repeat=False

   )


   plt.tight_layout()

   plt.show()

   return ani


animate_top10("batting_score", "Batsmen")

animate_top10("bowling_score", "Bowlers")

animate_top10("keeping_score", "Wicketkeepers")

animate_top10("final_score", "Overall Players")


def print_top(scores_col, title):

   print(f"TOP 10 {title}")

   print(df.sort_values(scores_col, ascending=False).head(10)[[

               'Player_Name', 'Role', scores_col, 'performance_score',
'predicted_class'

   ]])



print_top("batting_score", "BATSMEN")

print_top("bowling_score", "BOWLERS")

print_top("keeping_score", "WICKETKEEPERS")

print_top("final_score", "OVERALL PLAYERS")
```

12.3 NBA Dataset

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import matplotlib.animation as animation

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.svm import SVC

from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

from sklearn.model_selection import train_test_split


#Load Data

file_path = "NBA_Dataset.csv"

df = pd.read_csv(file_path)

df.columns = [c.strip() for c in df.columns]


metric_cols = ['PTS', 'AST', 'TRB', 'STL', 'BLK', 'FG%']


def clean(x):

   if pd.isna(x): return 0

   if isinstance(x, str):

       x = x.replace('%', '').replace(',', '')
```

```
        return float(x) if x.replace('.', '').isdigit() else 0

    return float(x)


for col in metric_cols:

    df[col] = df[col].apply(clean)


# Convert FG%

if df['FG%'].max() <= 1:

    df['FG%'] *= 100


norm = MinMaxScaler()

df_norm         =         pd.DataFrame(norm.fit_transform(df[metric_cols]),
columns=metric_cols)

df['performance_score'] = df_norm.mean(axis=1) * 100


q1, q2 = df['performance_score'].quantile([0.33, 0.66])

df['rating_class'] = df['performance_score'].apply(

    lambda x: 'Low' if x <= q1 else ('Medium' if x <= q2 else 'High')

)


X = df[metric_cols]

y = df['rating_class']


X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.3, stratify=y, random_state=42

)


sc = StandardScaler()

X_train_s = sc.fit_transform(X_train)

X_test_s = sc.transform(X_test)


svm = SVC(kernel='rbf', C=10, gamma='scale')

svm.fit(X_train_s, y_train)
```

```
df['predicted_class'] = svm.predict(sc.transform(X))


df['predicted_score'] = df['predicted_class'].map({'Low':20, 'Medium':60,
'High':100})

df['final_score'] = 0.7*df['performance_score'] +
0.3*df['predicted_score']


print("\nSVM CLASSIFICATION REPORT:")

y_pred = svm.predict(X_test_s)

print(classification_report(y_test, y_pred))


cm = confusion_matrix(y_test, y_pred)

print("\nCONFUSION MATRIX:")

print(cm)


print(f"\nACCURACY: {accuracy_score(y_test, y_pred)*100:.2f}%")


plt.figure(figsize=(7,5))

sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")

plt.title("Confusion Matrix")

plt.show()


def plot_nba_distributions():

   for col in metric_cols:

       plt.figure(figsize=(10,5))

       sns.histplot(df[col], kde=True, bins=20)


       avg = df[col].mean()

       max_val = df[col].max()


       sorted_vals = df[col].sort_values(ascending=False).values

       if len(sorted_vals) >= 10:
           top10_cut = sorted_vals[9]
       else:
           top10_cut = sorted_vals[-1]

       plt.axvline(avg, color='blue', linestyle='--', linewidth=2,
                   label=f"Average = {avg:.2f}")

       plt.axvline(top10_cut, color='red', linestyle='--', linewidth=2,
                   label=f"Top 10 Cutoff = {top10_cut:.2f}")

       plt.axvline(max_val, color='green', linestyle='--', linewidth=2,
                   label=f"Best Player (Max) = {max_val:.2f}")

       plt.title(f"{col} Distribution")
       plt.xlabel(col)
       plt.ylabel("Frequency")
       plt.legend()
       plt.tight_layout()
       plt.show()

print("\n Generating NBA parameter distribution graphs...")
plot_nba_distributions()

def pos_map(p):
   p = str(p).upper()
   if any(x in p for x in ['PG','SG','G']): return 'Guard'
   if any(x in p for x in ['SF','PF','F']): return 'Forward'
   if 'C' in p: return 'Center'
   return 'Unknown'

df['Group'] = df['Pos'].apply(pos_map)
df['Player Name'] = df['Player']
```

```python
def animate_top10(group, title):

    sub = df if group == 'All' else df[df['Group']==group]

    top10 = sub.sort_values('final_score', ascending=False).head(10)


    if top10.empty:

        print(f"No players found: {group}")

        return


    top10 = top10.sort_values('final_score')


    fig, ax = plt.subplots(figsize=(10,6))

    bars = ax.barh(top10['Player Name'], [0]*len(top10))


    ax.set_xlim(0, 100)

    ax.set_title(f"Top 10 {title}")


    def update(frame):

        ratio = frame/100

        for bar, score in zip(bars, top10['final_score']):

            bar.set_width(score * ratio)

        return bars


    ani = animation.FuncAnimation(

        fig, update,

        frames=np.linspace(0,100,60),

        interval=25,

        blit=False,

        repeat=False

    )


    plt.tight_layout()

    plt.show()

    return ani
```

```
animate_top10('Guard', "Guards")

animate_top10('Forward', "Forwards")

animate_top10('Center', "Centers")

animate_top10('All', "Overall Players")




print("TOP 10 GUARDS")

print(df[df['Group']=='Guard'].sort_values('final_score',ascending=False)
.head(10)[[

    'Player Name','final_score','performance_score','predicted_class'

]])


print("TOP 10 FORWARDS")

print(df[df['Group']=='Forward'].sort_values('final_score',ascending=Fals
e).head(10)[[

    'Player Name','final_score','performance_score','predicted_class'

]])


print("TOP 10 CENTERS")

print(df[df['Group']=='Center'].sort_values('final_score',ascending=False
).head(10)[[

    'Player Name','final_score','performance_score','predicted_class'

]])


print("TOP 10 OVERALL NBA PLAYERS")

print(df.sort_values('final_score', ascending=False).head(10)[[

                                                                  'Player
Name','Group','final_score','performance_score','predicted_class'

]])
```

References

1. Kumar, R., & Singh, A. (2021). *Support Vector Machine-based Cricket Talent Identification Model (SVMCTI).* International Journal of Computer Applications in Sports Analytics, 9(3), (Placeholder1)45–52.

2. Wong, T., & Chen, H. (2017). *Support Vector Machine-Based Prediction System for Football Match Result.* Journal of Sports Data Science, 5(2), 34–40.

3. Vapnik, V. N., & Cortes, C. (1995). *Support Vector Machines for Classification and Regression.* Machine Learning, 20(3), 273–297.

4. Li, Y., Zhang, Q., & Wang, S. (2020). *Forecasting Sports Injuries with Machine Learning.* Journal of Health Informatics and Analytics, 12(4), 210–218.

5. M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, Berkeley, CA: Apress, 2015.

6. P. Y. Lakshmi, S. Sanjaykumar, M. Dharuman, and A. Elangovan, *"Using Support Vector Regression Kernel Models for Cricket Performance Prediction in the Women's Premier League 2024," Physical Education Theory and Methodology*, vol. 24, no. 1, pp. 72–80, Feb. 2024.

7. S. Han, Q. Cao, and M. Han, *"Parameter Selection in SVM with RBF Kernel Function," IEEE Conference Publication*, Changchun University of Technology, China, 2014.

8. S. Suthaharan, *Machine Learning Models and Algorithms for Big Data Classification*, New York: Springer, 2016.

9. H. Liu, *"Technology-based Sports Performance Enhancement: A Conference Review," Intelligent Sports and Health*, vol. 1, pp. 98–102, 2025.

10. T. Evgeniou and M. Pontil, *"Workshop on Support Vector Machines: Theory and Applications,"* Center for Biological and Computational Learning, MIT Artificial Intelligence Laboratory, Cambridge, MA, USA, 1999

11. Zhang, X., & Wu, B. (2022). *Sports Performance Prediction Based on Support Vector Machine.* In *Proceedings of the International Conference on Artificial Intelligence and Intelligent Information Processing (AIIIP 2022)*, Qingdao, China. SPIE, Vol. 12456, Paper 124561T.