# Cricket player performance prediction

## Table of Contents

# Abstract

This project aims to enhance the accuracy of cricket match predictions by including pitch reports as an attribute in the dataset. Surprisingly, the cricket pitch holds significant sway over match outcomes, shaping the dynamics of the game and impacting both batting and bowling lineups. A well-prepared pitch with a flat surface is crucial for cricket. The characteristics of the pitch directly influence gameplay, favoring certain bowling or batting styles. For instance, a dry and dusty pitch provides better grip for spin bowling, giving an advantage to teams with skilled spin bowlers. By considering the pitch report as an attribute, the predictive model takes this critical factor into account, resulting in improved accuracy. It is essential to acknowledge that pitch conditions can vary from match to match, introducing a unique element to each game. By incorporating pitch reports in the dataset, the model acknowledges this significant aspect, enabling more informed predictions based on the specific characteristics of the playing surface. Including pitch reports enhances prediction accuracy by accounting for this influential variable and providing valuable insights for cricket match analysis and decision-making.

# Chapter - I
# INTRODUCTION

## 1.1 Existing System

**An efficient team prediction for one-day international matches using a hybrid approach of CS-PSO and machine learning algorithms**

Player classification is a critical aspect of cricket as it assists the coach and skipper in determining the roles of individual players in the squad and assigning tasks accordingly. It involves using performance statistics to classify players into categories such as batsmen, bowlers, batting allrounders, bowling allrounders, and wicketkeepers. This study presents an enhanced model for cricket team selection, employing an unbiased technique.

The selection process takes into consideration player performance, including batting average, bowling average, and an analysis of the opposing team's strengths and weaknesses. To improve the accuracy of machine learning prediction models, nature-inspired algorithms are utilized for feature optimization. Specifically, a hybrid approach combining CS-PSO (Cuckoo search -Particle swarm optimization) feature optimization and Support Vector Machine is employed.

The accuracy rates achieved using this approach are as follows: 97.14% for batsmen, 97.04% for bowlers, 97.28% for batting allrounders, 97.29% for bowling allrounders, and 92.63% for wicketkeepers.

By leveraging machine learning and feature optimization techniques, this enhanced model provides a robust framework for player classification in cricket. It ensures that players are selected based on their performance metrics, enabling the coach and skipper to make informed decisions when assigning roles and tasks within the team.

### 1.1.1 Limitations in Existing System

1. **Subjectivity in Player Classification:** Classifying players into specific categories, such as batsmen, bowlers, allrounders, and wicketkeepers, can be subjective to some extent. effectiveness in certain positions.
2. **Contextual Factors:** The model may not fully capture contextual factors such as pitch conditions, weather, team strategies, or player injuries. These factors can significantly impact a player's performance and their suitability for specific roles within the team.
3. **Scope of Feature Optimization:** While the hybrid CS-PSO and Support Vector Machine approach is employed for feature optimization, there may be other techniques or combinations of algorithms that could potentially enhance accuracy further. Exploring alternative approaches could help in improving the model's performance.

## 1.2 Proposed System

In this project, the inclusion of the pitch report as an attribute in the dataset enhances the prediction accuracy. The cricket pitch, surprisingly, holds significant influence over the outcome of a cricket match. It plays a pivotal role in shaping the dynamics of the game, affecting both the batting and bowling lineups. A well-prepared pitch, with a flat surface at the center of the ground, is crucial for the game of cricket.

The characteristics of the pitch have a direct impact on the gameplay. For instance, a dry and dusty pitch favors spin bowling due to the increased grip the ball gets on such surfaces. Consequently, teams with skilled spin bowlers gain an advantage when playing on such pitches. By considering the pitch report as an attribute in the dataset, the model accounts for this crucial factor and improves the accuracy of predictions.

It is important to acknowledge that the nature of the pitch can vary from match to match, introducing a unique element to each game. By incorporating the pitch report into the dataset, the model takes into account this significant aspect, enabling more informed predictions based on the specific characteristics of the playing surface.

### 1.2.1 Advantages over Existing System

1. **Enhanced Strategic Decision-Making:** Pitch conditions play a pivotal role in determining optimal strategies for a cricket match. Analyzing pitch report data empowers teams to make informed decisions regarding team composition, batting order, and bowling tactics. This enables them to adapt their game plans, leveraging the strengths and weaknesses associated with the specific pitch conditions.
2. **Improved Prediction Accuracy:** The inclusion of pitch report data as an attribute in the dataset enhances prediction accuracy. By considering the specific characteristics of the playing surface, the models can provide more precise predictions, accounting for the impact of the pitch on batting and bowling performances.
3. **Data-Driven Decision Making:** Including pitch report data ensures that decisions regarding team selection, batting order, and bowling plans are based on objective information. It reduces reliance on subjective opinions and enables teams to make data-driven decisions, enhancing overall team performance.

# Chapter II

# Literature Survey

In cricket, several academics have focused on predicting the team. A few of them are discussed in the following section.

V.S. Vetukuri et al. introduced a hybrid approach to enhance player selection by combining recurrent neural networks (RNN) and genetic algorithms (GA). In this methodology, the historical statistics of individual players are appropriately processed, and an initial feature grid is created for each player using a mathematical function within the GA. This improved feature matrix is then fed into the RNN, which calculates a final score for each player. The proposed approach generates a concurrent rank table, enabling team selectors to make efficient and precise player choices for upcoming matches.

F. Ahmed et al. proposed a multi-objective approach utilizing the NSGA-II algorithm to identify team members and optimize the overall batting and bowling strength. To evaluate batting and bowling performance, the researchers consider a player's batting average and bowling average in international T-20 cricket. After formulating the problem, they apply the elite non-dominated sorting genetic algorithm (NSGA-II) to conduct multi-objective genetic optimization for the entire team. To address various concerns, the authors use the knee region analysis to generate feasible solutions, which are then assessed based on fitness scores across all relevant metrics. This comprehensive approach ensures that both batting and bowling strengths are optimized while considering the trade-offs and complexities associated with selecting the most effective team members.

A. Khot et al. proposed a novel to identify promising cricket players and enhance team selection through the concept of co-players. In this approach, a support vector machine (SVM) was trained as a discriminative classifier for classification purposes. By plotting a hyperplane, the algorithm divides data points into two groups: (1) those identified as rising stars and (2) those not classified as rising stars. The RS score, which measures the favorable and adverse characteristics of co-players, was utilized to compile the final list of rising stars in the batting domain, achieving an accuracy of 60%. For the bowling domain, the accuracy reached 70%, while for all-rounders, the assessment accuracy was 40%. This innovative method provides a valuable framework for identifying emerging talents and improving team selection processes in cricket.

I. Wickramasinghe presents a categorization of all-rounders based on the mean of their batting and bowling averages. The author identifies four types of all-rounders: genuine all-rounders, batter all-rounders, bowling all-rounders, and ordinary all-rounders. To make predictions, the study employed the Naive Bayes (NB), K-Nearest Neighbors (KNN), and Random Forest (RF) algorithms. The k-fold cross-validation method was utilized to evaluate the performance of these analytical approaches. The experimental results indicate that the Random Forest (RF) algorithm exhibits significantly higher prediction accuracy compared to the other methods.

A. Balasundaram et al. employed K-means clustering, decision trees, support vector machine (SVM), and random forest algorithms for player classification. The data mining program WEKA was utilized to perform operations on the provided dataset. Cross-fold validation was employed to generate training and testing data from class-labelled data. The classifier achieved a prediction accuracy of 91.87% using the decision tree, 93.46% with SVM, and 95.78% using the random forest algorithm. These results indicate that the developed model was effective in accurately predicting the best player option for the team.

M. Bello et al. presented an innovative approach to team formation, wherein two organizations collaborate to build teams by selecting individuals from a shared pool of applicants. This research introduces the Ant Colony Optimization (ACO) and the Max-Min Ant System to Team Formation (MMAS-TS) metaheuristics. The solutions discovered by the best pair of ants in each iteration exhibit different structures. The pair of ants that achieves the most effective solution throughout the execution receives a pheromone deposit. Each decision-maker evaluates the applicants based on their preferences for team formation while considering the overall quality.

B. Kapadiya et al. aimed to forecast the number of runs a batter would score and the number of wickets a bowler would take in a specific game on a particular day. To achieve this, they utilized machine learning techniques, including decision trees, support vector machine (SVM), naïve Bayesian, and random forest algorithms for prediction. The researchers introduced a methodology that incorporates a weather dataset along with cricket match information to predict player performance. Their model employs a novel weighted random forest classifier with hyperparameter tuning, achieving an accuracy of 92.25%.

P. Chhabra et al. presented a method for constructing a team selection by modeling players into embeddings using a semi-supervised statistical technique. The article introduces a grading system called the 'Quality Index of Player' that takes into account both qualitative and quantitative factors for evaluating players. The researchers developed a semi-supervised team suggestion framework called CRICTRS, which utilizes player embeddings to recommend a team based on the opponent's strengths and weaknesses. This approach is derived from collaborative filtering and the Bernoulli experiment, which ranks players based on the quality of their runs and wickets.

H. Ahmad et al. utilized supervised machine learning models to forecast Star Cricketers based on their performance in both batting and bowling domains. Bayesian rule functions and decision-tree-based frameworks were employed for making predictions, and the performance was validated using cross-validation. The importance of each feature in the prediction task was determined using advanced metrics such as information gain, gain ratio, and chi-squared statistics. In the batting domain, the Support Vector Machine (SVM) yielded remarkable results with an accuracy of 86.59%. For the bowling dataset, the Naïve Bayes (NB) and CART models outperformed other models with an accuracy of 88.9%.

Q. Agrawal and T. Ganesh describe the player selection process of the Indian cricket team using integer optimization programming. The efficiency of a player for a match is determined based on two variables in batting statistics: the batter's average and their consistency. For bowlers, their performance is evaluated using factors such as bowling average, consistency, and economy rate. Spearman's rank correlation coefficient is employed to assess the typical technique of varied values. Integer programming is utilized with a decision variable of unit function type to select the cricket team. The player's consistency played a crucial role in the team's rating.

Z. Mahmood et al. addressed the prediction of emerging basketball stars as a machine learning problem by employing various algorithms, including Classification and Regression Trees, Support Vector Machines, Maximum Entropy Markov Model, Bayesian Network, and Naïve Bayes. Their objective was to develop a function capable of predicting whether a player belongs to the category of rising stars or not based on a set of features. To overcome the absence of an average efficiency score for each co-player, they incorporated the h-index as a compensatory measure. The Hollinger linear formula was utilized to calculate the efficiency of co-players. A 10-fold cross-validation approach was applied to train and evaluate the algorithms using three distinct datasets, thereby assessing the robustness of the classifiers.

I. Kumarasiria and S. Perera employed a genetic algorithm to determine the optimal cricket team composition. The players' fitness levels were evaluated based on factors such as batting average, batting strike rate, bowling average, total wickets taken in each match, win percentage, and experience. The overall fitness of each player was calculated by assigning fitness values to their respective chromosomes. The fitness values for each team were derived using the mutation approach. The algorithm continued iterating until the difference between the fitness scores of the best and worst players became minimal, thus serving as the stopping condition for the optimization process.

M. Ishi and J. Patil conducted a comprehensive review of team formation and winner prediction techniques in cricket. They extensively examined the various parameters and methodologies employed for player evaluation. Through their study, they identified parameters and procedures that can be utilized for team formation. It was observed that each parameter carries its own significance and, accordingly, the weighting of these parameters needs to be determined for effective decision-making.

# Chapter III

# Software Requirement Specifications

## Feasibility Study

### 3.1 Technical Feasibility

The technical feasibility of cricket player performance prediction using machine learning techniques is quite high. Here are some key factors that contribute to its feasibility.

1. **Availability of Data:** Cricket generates a significant amount of data, including historical match statistics, player performance records, pitch conditions, weather information, and more. This data can be leveraged to train machine learning models and make accurate predictions about player performance.
2. **Data Collection and Integration**: Although data collection can be a challenging task, various sources provide comprehensive cricket data, including cricket boards, sports organizations, and online platforms. With proper data integration and preprocessing techniques, it is feasible to gather relevant data for training and prediction purposes.
3. **Machine Learning Algorithms**: Machine learning algorithms, such as regression models, decision trees, random forests, support vector machines (SVM), and neural networks, have been successfully applied in sports analytics, including cricket player performance prediction. These algorithms can handle complex patterns in the data and make accurate predictions based on historical trends and player attributes.
4. **Computational Power**: The computational power required for training and running machine learning models has significantly improved with advancements in hardware, cloud computing, and distributed computing frameworks. This makes it feasible to handle large datasets and complex models efficiently.
5. **Feature Engineering and Selection:** Feature engineering and selection play a crucial role in extracting meaningful information from the available data. With the expertise of domain specialists and data scientists, relevant features can be identified and engineered to capture the key factors influencing player performance.
6. **Model Evaluation and Validation:** Rigorous evaluation and validation techniques, such as cross-validation, performance metrics analysis, and comparison with ground truth data, can ensure the reliability and accuracy of the predictive models. This helps in assessing the feasibility of the predictions and identifying areas for improvement.
7. **Integration with Existing Systems:** Cricket player performance prediction models can be integrated with existing cricket analytics systems, coaching tools, and team management software. This allows coaches, selectors, and team managers to leverage the predictions and make data-driven decisions in team selection, strategy formulation, and performance analysis.

### 3.2 Economic Feasibility

The economic feasibility of cricket player performance prediction using machine learning techniques depends on several factors. Here are some considerations for assessing the economic feasibility:

1. **Cost of Data Acquisition:** The cost associated with acquiring cricket data, including player statistics, match records, and other relevant data sources, needs to be taken into account. This can involve purchasing data from providers, accessing APIs, or investing in data collection infrastructure. The availability of free or low-cost data sources can help reduce the economic burden.
2. **Computational Resources:** Training and running machine learning models often require substantial computational resources, including powerful hardware, cloud computing services, and infrastructure. The economic feasibility depends on the cost of acquiring and maintaining these resources, as well as the efficiency of the models in terms of computation time and resource utilization.
3. **Expertise and Manpower:** Developing and implementing cricket player performance prediction models requires skilled data scientists, machine learning experts, and domain specialists. The economic feasibility depends on the availability and cost of hiring or training such personnel. Collaborations with research institutions or partnerships with experts in the field can also help reduce costs.
4. **Return on Investment:** The economic feasibility is determined by the potential return on investment (ROI) from using cricket player performance prediction models. This can include benefits such as improved team selection, enhanced player development, optimized strategies, and increased team performance. The value gained from these improvements needs to outweigh the costs associated with developing and maintaining the prediction system.
5. **Commercialization Opportunities:** Cricket player performance prediction models can have commercialization potential in areas such as sports betting, fantasy leagues, sports analytics services, and player scouting. Exploring these opportunities can contribute to economic feasibility by generating revenue streams and attracting potential investors or sponsors.
6. **Cost Savings and Efficiency:** The economic feasibility can be justified if the predictive models lead to cost savings or improved efficiency in player selection processes, training programs, and resource allocation. By making data-driven decisions, teams and cricket organizations can potentially reduce expenses related to scouting, trial sessions, and player recruitment.
7. **Market Demand:** Assessing the market demand for cricket player performance prediction solutions is crucial for economic feasibility. Understanding the needs of cricket teams, coaches, selectors, and sports enthusiasts will help determine the potential value and demand for such predictive models.

### 3.3 Operational feasibility

The operational feasibility of cricket player performance prediction refers to assessing the practicality and effectiveness of implementing such a system within the operational environment. Several factors need consideration to evaluate the operational feasibility:

1. **Data Availability and Quality:** Access to accurate and reliable player statistics, match records, and relevant data sources is crucial. Ensuring data quality and timeliness is vital for the success of the prediction system.
2. **Technical Infrastructure:** Assessing the existing technical infrastructure's capacity to support the prediction system is important. This includes evaluating computational resources, storage capacity, and network infrastructure.
3. **Integration with Existing Systems:** The system should integrate smoothly with existing cricket management systems, data platforms, or analytics tools.
4. **User Acceptance and Adoption:** User acceptance and adoption by coaches, selectors, and team management are critical. Involving stakeholders and gathering feedback during development ensures the system meets operational requirements.
5. **Training and Support:** Adequate training and support should be provided to end users to ensure successful implementation and operation.
6. **Legal and Ethical Considerations:** Compliance with data privacy, security, and ethical guidelines is essential.
7. **Scalability and Maintenance:** The system's scalability to handle growing data volumes and user demands, as well as maintenance requirements for software updates and bug fixes, should be assessed.

# Chapter IV

# Design

## 4.1 Project description

The project's objective is to develop an advanced system using machine learning that can accurately predict the performance of cricket players across different game formats. By analyzing historical data and applying statistical techniques, the system will provide valuable assistance to coaches, selectors, and team management in making informed decisions related to player selection, team composition, and strategic planning.

The project will encompass the following key components:

1. **Data Collection and Processing:** A comprehensive dataset comprising historical player performance data, match statistics, pitch conditions, team compositions, and other pertinent factors will be gathered. Thorough data preprocessing techniques will be employed to handle missing values, outliers, and inconsistencies.
2. **Feature Selection and Engineering:** Essential features will be identified and extracted from the dataset to represent player performance. These features may include batting average, bowling average, strike rate, economy rate, player rankings, and various statistical measures. Contextual factors such as the strength of the opposing team, playing conditions, and match format will also be considered to capture relevant information.
3. **Development of Machine Learning Models:** Various machine learning algorithms and techniques, such as regression models, decision trees, random forests, and neural networks, will be explored to develop a predictive model. The model will be trained using historical data, and appropriate evaluation metrics will be employed to assess its performance.
4. **Performance Prediction:** The trained model will be utilized to predict the performance of cricket players in upcoming matches or tournaments. The system will generate insights and predictions regarding the batting average, bowling average, player rankings, and other performance metrics. These predictions will be based on the analysis of individual player characteristics, match context, and historical trends.

### 4.1.1 System Use Case Diagrams

Use case diagrams consist of use cases, actors, and relationships. Use cases represent the specific tasks or actions that users can perform within the system. Actors, on the other hand, represent the external entities (users or systems) that interact with the system.

In a use case diagram, use cases are depicted as ovals or ellipses, while actors are represented as stick figures or other appropriate symbols. The relationships between actors and use cases are shown using lines or arrows.

Use case diagrams serve several purposes:

1. **Requirement Analysis**: Use case diagrams help in understanding the system's requirements by identifying the various use cases and the actors involved. They provide a clear overview of the system's functionalities and how users or external systems interact with it.
2. **Communication:** Use case diagrams facilitate effective communication between stakeholders, including developers, designers, testers, and users. They provide a common understanding of the system's behavior and functionality.
3. **System Design:** Use case diagrams aid in system design by identifying the major components, modules, or subsystems that are necessary to support the identified use cases. They help in organizing and structuring the system architecture.
4. **Test Planning:** Use case diagrams assist in test planning by identifying the different scenarios or test cases that need to be covered for each use case. They help in ensuring comprehensive test coverage and validating the system's behavior.
5. **System Documentation:** Use case diagrams serve as a documentation tool, providing a visual representation of the system's functionality. They can be used as a reference for developers, maintainers, or users to understand the system's intended behavior.
6. **Project Management:** Use case diagrams aid in project management by identifying the scope and boundaries of the system. They assist in estimating project timelines, allocating resources, and managing the development process.

**Bowler data acquisition**

The use case diagram comprises two elements: Agent1 and System. Agent1 is tasked with gathering cricket data in YAML format and supplying it to the bowler data acquisition system. The system then converts the YAML data into a dictionary format, ensuring the data maintains a structured order. The dictionary data is subsequently converted into Dataframes, which are tabular representations suitable for subsequent computations. To facilitate long-term storage and future retrieval, the Dataframes are stored in Excel sheets. The data is stored in two distinct formats: match bowler data and overall bowler data.

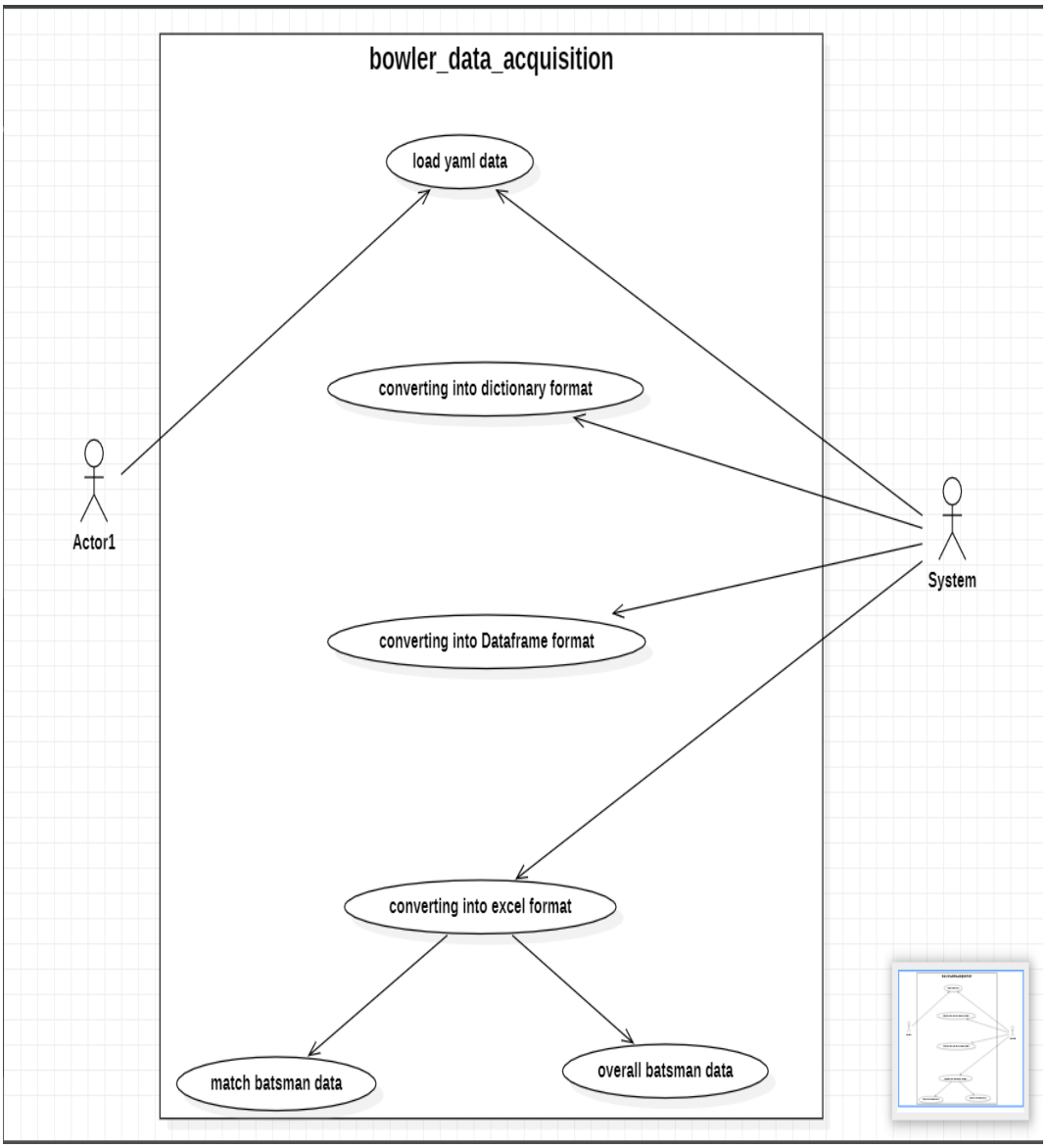


**Fig.1 Use case diagram for Bowler data acquisition**

**Batsman data acquisition**

The use case diagram involves two entities: Agent1 and System. Agent1 is responsible for collecting cricket data in YAML format and providing it to the bowler data acquisition system. The system then converts the YAML data into a dictionary format, ensuring the data is in an ordered structure. Subsequently, the dictionary data is transformed into Dataframes, which are tabular representations suitable for further computation. To ensure long-term storage and future reference, the Dataframes are stored in Excel sheets. The data is stored in two formats: match batsman data and overall batsman data.

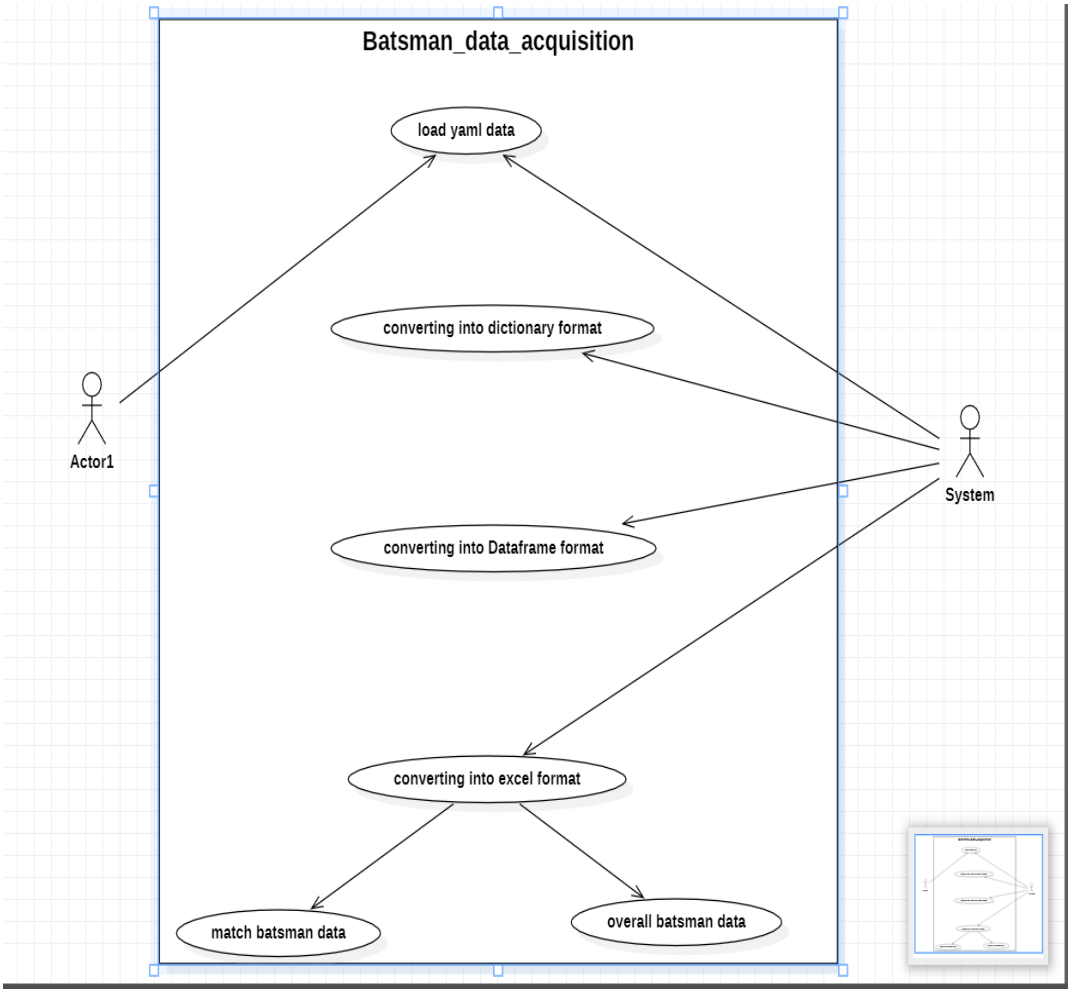


**Fig.2 Use case diagram for Batsman data acquisition**

**Player performance**

In this use case, there are three agents involved: batsman data acquisition, bowler data acquisition, and Machine Learning Model. The batsman data acquisition and bowler data acquisition agents load data into the System, which is stored in the form of Excel sheets. The data then undergoes Machine Learning techniques for data preprocessing, such as Feature Extraction, where attributes with high predictive power are selected for the Model. The Model is constructed using a supervised Machine Learning technique called RandomForest, which is suitable for both Classification and Regression problems. Once the Model is built, it is trained and tested iteratively until it produces improved results.

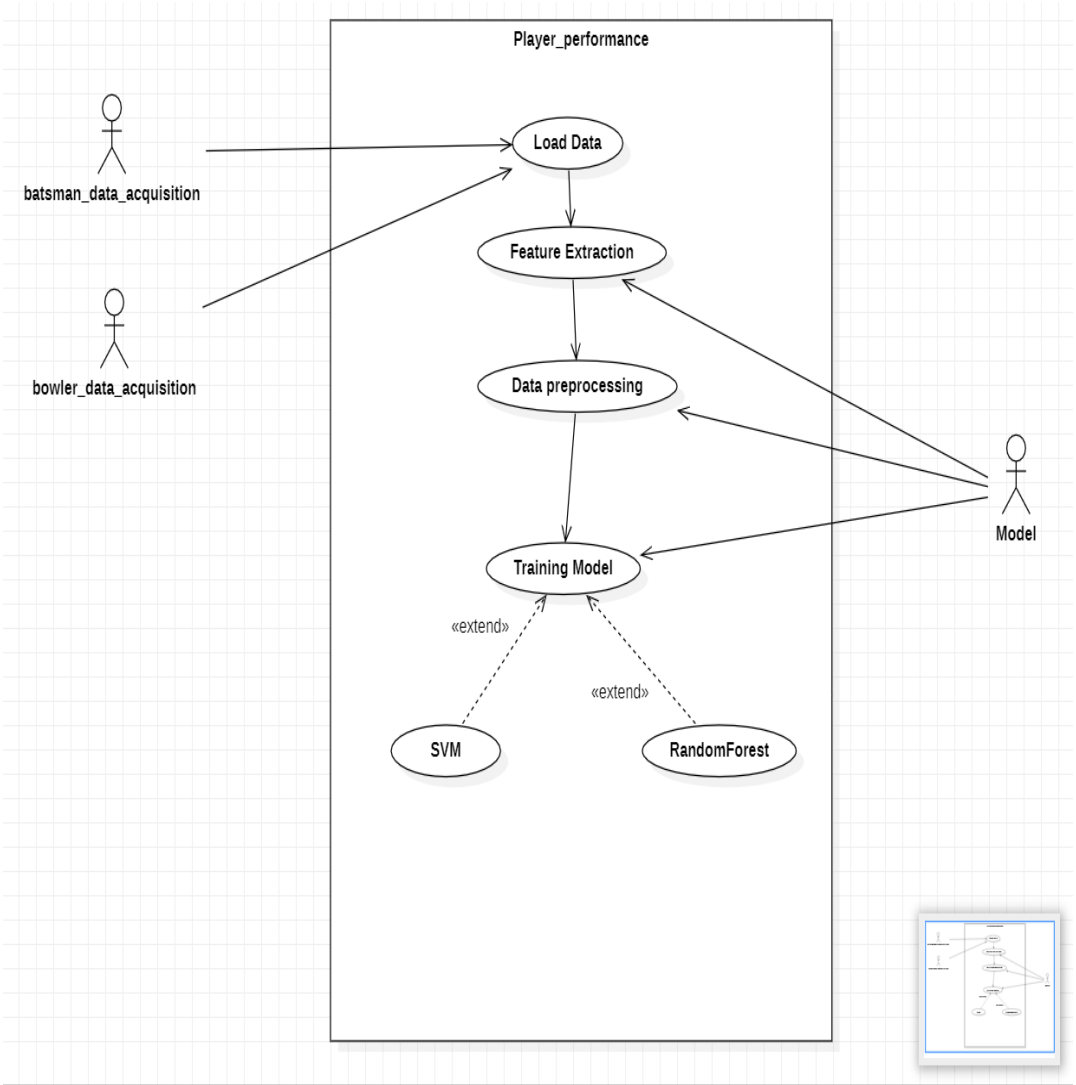


**Fig.3 Use case diagram for Player performance**

## Prediction

In this particular use case, two agents are involved: User and Model. The system is connected to a Player Performance System where the model is stored and trained. When the user runs the file, an interface is presented. The model prompts the user for specific information, such as the desired team, player name, opposition team, venue and pitch report. By considering these inputs, the model makes predictions regarding the number of wickets taken by the bowler and the number of runs scored by the batsman.

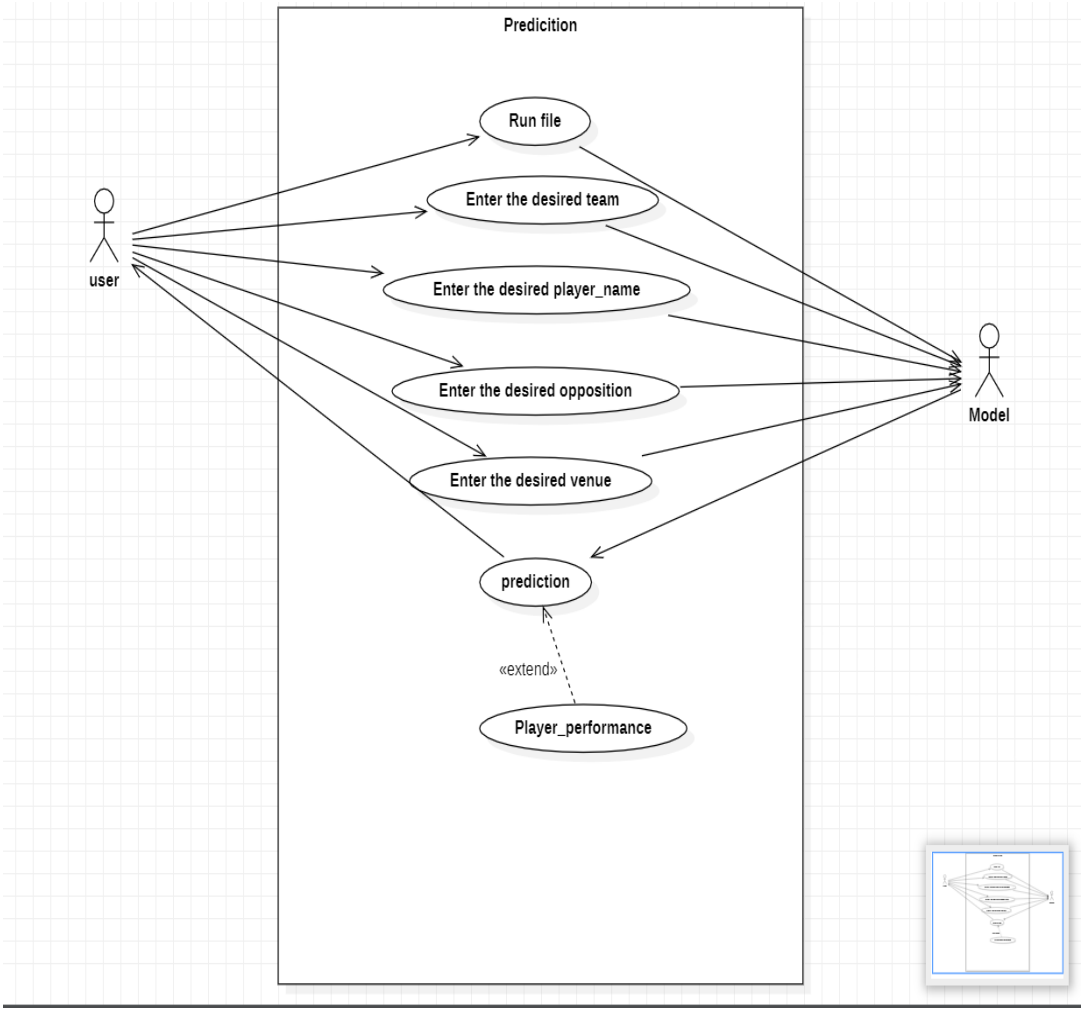


**Fig.4 Use case diagram for Prediction**

### 4.1.2 Class diagram

A UML (Unified Modeling Language) class diagram is a visual representation of the structure and relationships of classes in an object-oriented system. It provides a high-level overview of the system's architecture, showing the classes, their attributes, methods, and associations.
Here are some key elements commonly found in a UML class diagram:

**1. Class:** A class represents a blueprint for creating objects. It is depicted as a rectangle with three sections. The top section contains the class name, the middle section includes the class's attributes (variables), and the bottom section lists the class's methods (functions).

**2. Attributes:** Attributes represent the characteristics or data associated with a class. They are typically displayed as a name followed by a colon and the data type. For example, "name: String" indicates an attribute named "name" of type "String."

**3. Methods:** Methods represent the behavior or operations that can be performed by a class. They are typically displayed in the bottom section of the class rectangle and follow a similar notation as attributes. For example, "calculateTotal(): float" indicates a method named "calculateTotal" that returns a float value.

**4. Associations:** Associations represent relationships between classes. They indicate how classes are connected or interact with each other. Associations can be one-to-one, one-to-many, or many-to-many, and they are often annotated with multiplicity to denote the number of instances involved in the relationship.
**5. Inheritance:** Inheritance represents the "is-a" relationship between classes, where one class (child or subclass) inherits the properties and behaviors of another class (parent or superclass). Inheritance is denoted by an arrow pointing from the subclass to the superclass.
**6. Aggregation and Composition:** Aggregation and composition are special types of associations that represent part-whole relationships. Aggregation implies that the whole can exist independently of its parts, while composition implies that the parts are exclusive to the whole and cannot exist without it.
**7. Interfaces:** Interfaces define a contract for classes that implement them. They specify a set of methods that implementing classes must provide. Interfaces are depicted as a circle with the interface name, and classes that implement the interface are connected to it with a dashed line.

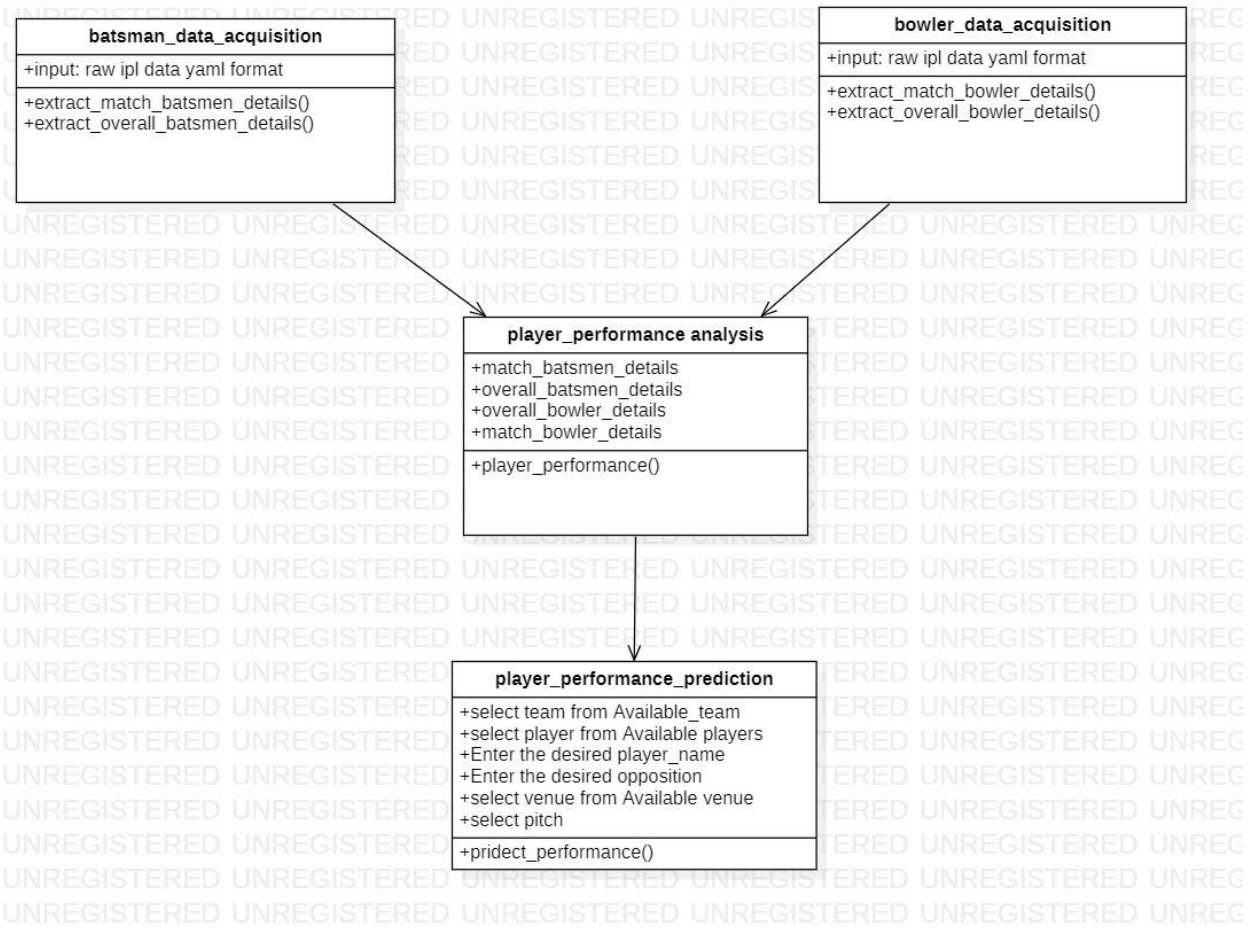


**Fig.5 Class diagram**

The UML class diagram consists of four classes: BatsmanDataAcquisition, BowlerDataAcquisition, PlayerPerformanceAnalysis, and PlayerPerformancePredictionIn the BatsmanDataAcquisition class, there are two methods: extractMatchBatsmanDetails and extractOverallBatsmanDetails. These methods retrieve batsman details from YAML files. Similarly, in the BowlerDataAcquisition class, there are two methods: extractMatchBowlerDetails and extractOverallBowlerDetails. These methods extract bowler data from YAML files. In the PlayerPerformanceAnalysis class, the playerPerformance method is executed. This involves creating a model and performing operations such as training and testing. Finally, in the PlayerPerformancePrediction class, the predict performance method takes user inputs and predicts the outcomes

## 5. Data Flow Diagram

A data flow diagram (DFD) is a graphical representation that shows how information moves within a system or process. It illustrates the flow of data among different entities, processes, and data storage locations. DFDs are commonly used in system analysis and design to provide a clear and concise understanding of how information is exchanged.
DFDs consist of several components:
**1. Processes:** Processes depict the actions or operations performed on the data. They are represented as circles or rectangles and describe specific functions. Processes take input data, perform actions, and produce output data.
**2. Data Flows:** Data flows indicate the movement of data within the system. They are depicted as arrows and show the direction of data transmission. Data flows represent the transfer of information between processes, entities, and data storage.
**3. Entities:** Entities represent external sources or recipients of data that interact with the system. They can be organizations, individuals, or other systems. Entities are typically depicted as rectangles and describe the origin or destination of data.
**4. Data Stores:** Data stores symbolize repositories or databases where data is stored or retrieved. They can be physical or virtual storage locations. Data stores are represented as rectangles and indicate the type of data they contain.
**5. Data Labels:** Data labels provide descriptions of the transmitted or stored data. They clarify the meaning or content of the data and facilitate understanding of its purpose within the system.

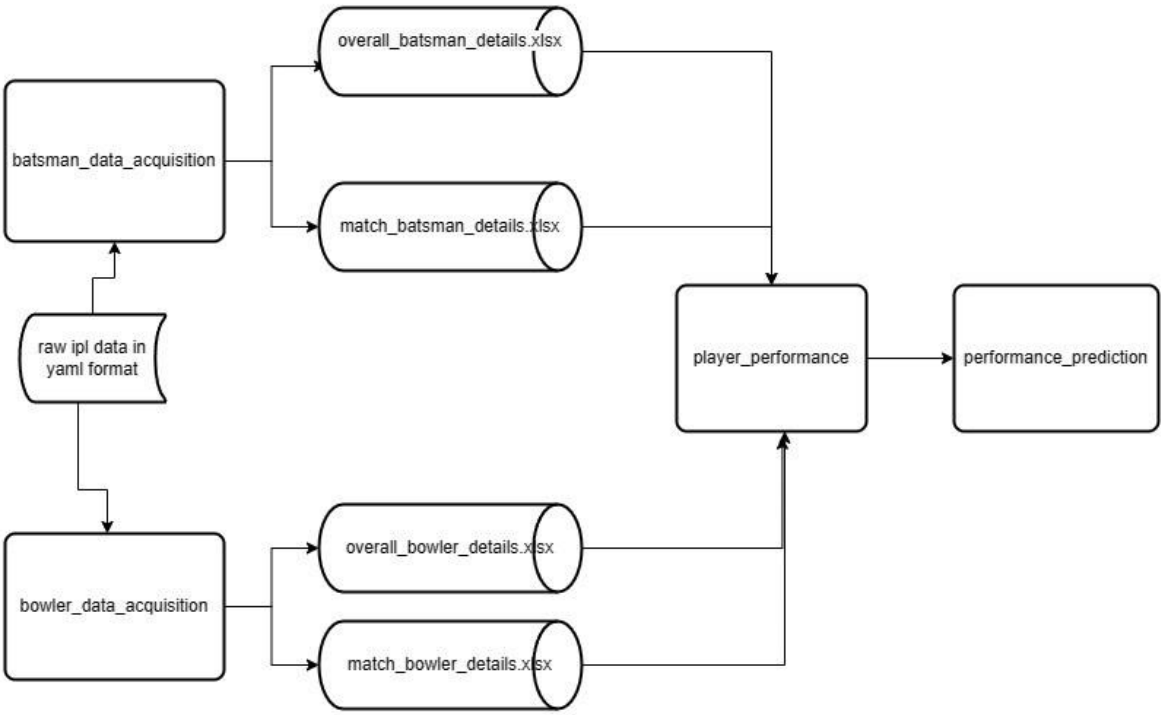


**Fig.6 Data Flow Diagram**

In this specific data flow diagram (DFD), the process involves extracting data from YAML files that contain cricket-related information. The extracted data is then transformed into an Excel

format. The Excel files resulting from this transformation are subsequently used as input for a model or system.

## Architecture:

In this architecture, we have 3 major parts. They are

- **Bowler data acquisition:** The extraction of bowlers overall and match data from YAML files. The extracted data is converted into an Excel file for further operations.
- **Batsman data acquisition:** The extraction of batsman overall and match data from YAML files. The extracted data is converted into an Excel file for further operations.
- **Player performance analysis**: The data is present in an Excel file which is obtained from previous operations. The data from an excel file is given input for the model. The model is based on RandomForest and SVM algorithms.

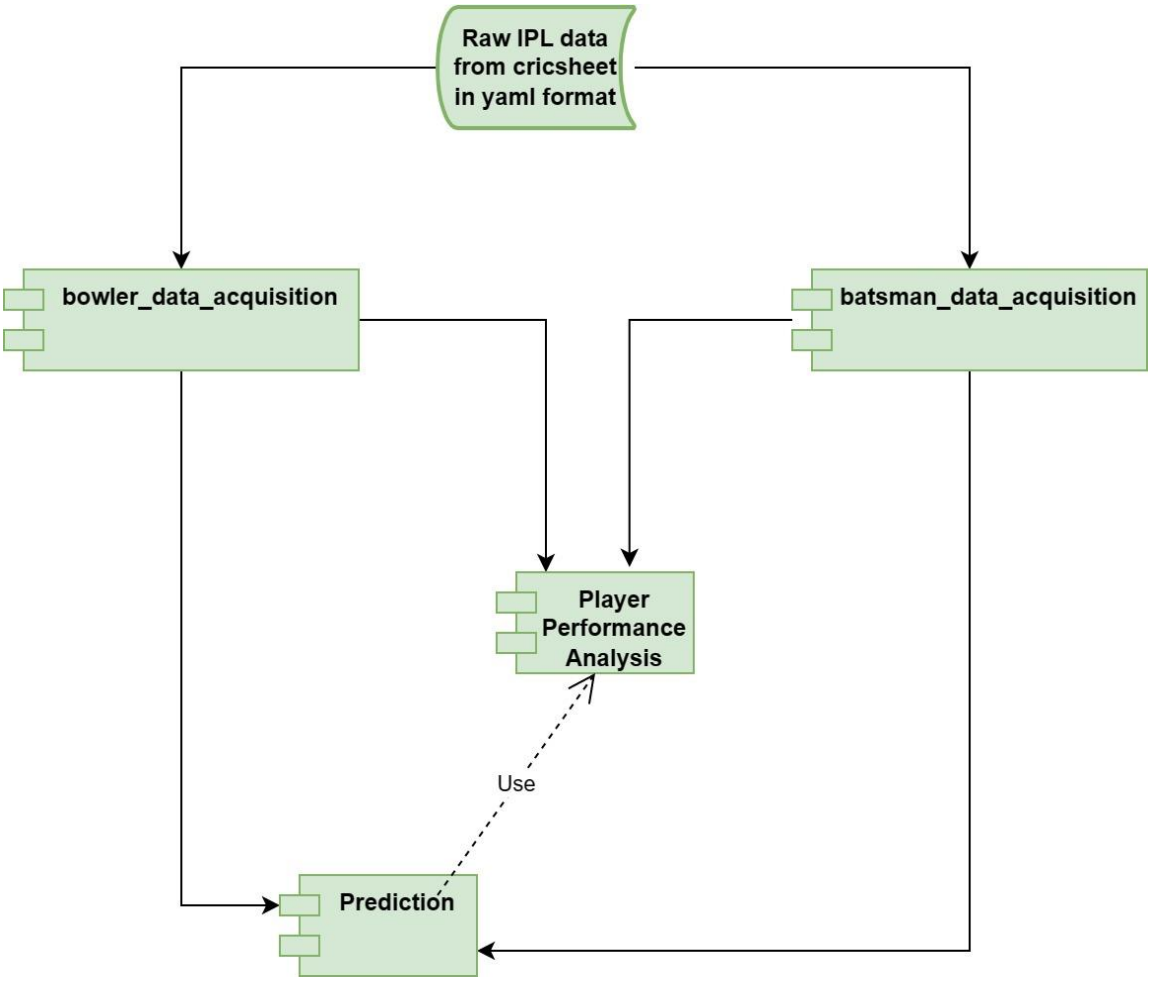


**Fig.7 Architecture diagram**

# Chapter V

# Implementation

## 5.1 Source Code

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC
#Ignoring XGBoost warnings
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn

#Ignoring SciKit-Learn warnings
import warnings
from sklearn.exceptions import DataConversionWarning
```

**Fig.8 Importing modules**

```
#Extracting Targets and Features
if param == 1:
    overall_batsman_details = pd.read_excel("C:\\Users\\SHIVA KUMAR\\OneDrive\\Documents\\Cricket player performance prediction2\\overall_batsman_details.xlsx", header=0)
    match_batsman_details = pd.read_excel("C:\\Users\\SHIVA KUMAR\\OneDrive\\Documents\\Cricket player performance prediction2\\match_batsman_details.xlsx",header=0)
    match_batsman_details.loc[:, 'date'].ffill(inplace=True)
    bat_match_details = match_batsman_details[match_batsman_details['name']==player_name]
    bat_match_details = bat_match_details[bat_match_details['opposition']==opposition]
    bat_overall_details = overall_batsman_details[overall_batsman_details['player_name'] == player_name][['player_name', 'team', 'innings', 'runs', 'average', 'strike_rate'
    bat_features = bat_match_details.loc[:,['opposition', 'venue', 'innings_played','previous_average', 'previous_strike_rate', 'previous_centuries','previous_fifties', 'pr
    bat_targets = bat_match_details.loc[:,['runs']]

elif param == 2:
    overall_bowler_details = pd.read_excel("C:\\Users\\SHIVA KUMAR\\OneDrive\\Documents\\Cricket player performance prediction2\\overall_bowler_details.xlsx",header=0)
    match_bowler_details = pd.read_excel("C:\\Users\\SHIVA KUMAR\\OneDrive\\Documents\\Cricket player performance prediction2\\match_bowler_details.xlsx",header=0)
    match_bowler_details.loc[:, 'date'].ffill(inplace=True)
    bowl_match_details = match_bowler_details[match_bowler_details['name']==player_name]
    bowl_match_details = bowl_match_details[bowl_match_details['opposition'] == opposition]
    bowl_overall_details = overall_bowler_details[overall_bowler_details['player_name']==player_name][['player_name','team','innings','wickets','average','strike_rate','ecc
    bowl_features = bowl_match_details.loc[:,['opposition', 'venue', 'innings_played','previous_average', 'previous_strike_rate', 'previous_economy','previous_wicket_hauls'
    bowl_targets = bowl_match_details.loc[:,['wickets']]

elif param == 3:
    overall_batsman_details = pd.read_excel("C:\\Users\\SHIVA KUMAR\\OneDrive\\Documents\\Cricket player performance prediction2\\overall_batsman_details.xlsx", header=0)
    match_batsman_details = pd.read_excel("C:\\Users\\SHIVA KUMAR\\OneDrive\\Documents\\Cricket player performance prediction2\\match_batsman_details.xlsx",header=0)
    match_batsman_details.loc[:, 'date'].ffill(inplace=True)
    bat_match_details = match_batsman_details[match_batsman_details['name']==player_name]
    bat_match_details = bat_match_details[bat_match_details['opposition'] == opposition]
    bat_overall_details = overall_batsman_details[overall_batsman_details['player_name']==player_name][['player_name','team','innings','runs','average','strike_rate','centu
    bat_features = bat_match_details.loc[:,['opposition', 'venue', 'innings_played','previous_average', 'previous_strike_rate', 'previous_centuries','previous_fifties', 'pr
    bat_targets = bat_match_details.loc[:,['runs']]
```

**Fig.9 Extracting of data**

```
        #FinalModeling
            #XGBoost
        if bat_best_score[1] == 'xgb':
            if classes_bat > 2:
                bat_classifier = XGBClassifier(objective='multi:softmax',n_estimators=bat_best_params['n_estimators'],learning_rate=bat_best_params['learning_rate'],booster
            else:
                bat_classifier = XGBClassifier(objective='binary:logistic',min_leaf_samples=1,n_estimators=bat_best_params['n_estimators'], learning_rate=bat_best_params['l
            bat_classifier = bat_classifier.fit(bat_features,bat_targets)
            res['bat_prediction'] = bat_classifier.predict(predict_bat_features)
            #RandomForestClassifier
        elif bat_best_score[1] == 'rfc':
            if classes_bat > 2:
                bat_classifier = RandomForestClassifier(n_estimators=bat_best_params['n_estimators'],criterion=bat_best_params['criterion'],random_state=42,min_samples_leaf
            else:
                bat_classifier = RandomForestClassifier(n_estimators=bat_best_params['n_estimators'],criterion=bat_best_params['criterion'],random_state=42,min_samples_leaf
            bat_classifier = bat_classifier.fit(bat_features,bat_targets)
            res['bat_prediction'] = bat_classifier.predict(predict_bat_features)
            #SupportVectorMachine
        elif bat_best_score[1] == 'svc':
            bat_classifier = SVC(C=bat_best_params['C'],kernel=bat_best_params['kernel'],gamma=bat_best_params['gamma'])
            bat_classifier = bat_classifier.fit(bat_features,bat_targets)
            res['bat_prediction'] = bat_classifier.predict(predict_bat_features)

        bat_runs = {'0':'0-10','1':'11-30','2':'31-50','3':'51-80','4':'81-120','5':'121-250'}
        res['bat_prediction'] = bat_runs[res['bat_prediction'][0]]

        print('Batting Prediction Ends!')

    else:

        print('NO Batting Prediction')
```

**Fig.10 Implementation of Model**

## 5.2 Results

```
Available Services:
1. Specific Player Performance


Available team : "      "Sunrisers Hyderabad"   "Royal Challengers Bangalore"   "Mumbai Indians"        "Rising Pune Supergian
t"      "Gujarat Lions" "Kolkata Knight Riders" "Kings XI Punjab"       "Delhi Daredevils"      "Chennai Super Kings"   "Rajast
han Royals"     "Delhi Capitals"        "Punjab Kings"  "Lucknow Super Giants"  "Gujarat Titans"        "Deccan Chargers"
"Kochi Tuskers Kerala"  "Pune Warriors" "Rising Pune Supergiants

Enter the desired team: Royal Challengers Bangalore
1 Royal Challengers Bangalore

Available player_name : "       "A Choudhary"   "A Kumble"      "A Mithun"      "AA Noffke"     "AB Dinda"      "AB McDonald"
"AF Milne"      "AN Ahmed"      "AUK Pathan"    "Abdur Razzak"  "Akash Deep"    "B Akhil"       "C de Grandhomme"       "CH Gay
le"     "CH Morris"     "CJ Anderson"   "CJ Jordan"     "CK Langeveldt" "CL White"      "CR Woakes"     "D Wiese"       "D du P
reez"   "DJ Willey"     "DJG Sammy"     "DL Vettori"    "DR Sams"       "DT Christian"  "DW Steyn"      "GHS Garton"    "GJ Max
well"   "HV Patel"      "I Udana"       "Iqbal Abdulla" "J Syed Mohammad"       "JA Morkel"     "JD Ryder"      "JD Unadkat"
"JH Kallis"     "JJ van der Wath"       "JR Hazlewood"  "KA Jamieson"   "KP Appanna"    "KP Pietersen"  "KW Richardson" "LRPL T
aylor"  "M Ashwin"      "M Kartik"      "M Muralitharan"        "MA Starc"      "MC Henriques"  "MK Lomror"     "MM Ali"
"MP Stoinis"    "Mohammed Siraj"        "NA Saini"      "P Kumar"       "P Negi"        "P Parameswaran"        "P Ray Barman"
"PWH de Silva"  "Pankaj Singh"  "Parvez Rasool" "R Ninan"       "R Rampaul"     "R Vinay Kumar" "RE van der Merwe"      "RR Bha
tkal"   "S Aravind"     "S Badree"      "S Dube"        "S Rana"        "S Sriram"      "SA Abbott"     "SB Joshi"      "SR Wat
son"    "STR Binny"     "Sachin Baby"   "Shahbaz Ahmed" "TG Southee"    "TM Dilshan"    "TM Head"       "TS Mills"      "UT Yad
av"     "V Kohli"       "VR Aaron"      "Washington Sundar"     "YS Chahal"     "Yuvraj Singh"  "Z Khan

Enter the desired player_name: V Kohli

Available opposition : "        "Chennai Super Kings"   "Deccan Chargers"       "Delhi Daredevils"      "Gujarat Lions" "Kings
XI Punjab"      "Kochi Tuskers Kerala"  "Kolkata Knight Riders" "Mumbai Indians"        "Rajasthan Royals

Enter the desired opposition: Chennai Super Kings

Available venue : "     "Dr DY Patil Sports Academy"    "Eden Gardens"  "Feroz Shah Kotla"      "Himachal Pradesh Cricket Assoc
iation Stadium" "M Chinnaswamy Stadium" "MA Chidambaram Stadium, Chepauk"       "Nehru Stadium" "New Wanderers Stadium" "Punjab
Cricket Association Stadium, Mohali"    "Rajiv Gandhi International Stadium, Uppal"     "Sardar Patel Stadium, Motera"  "Sawai
Mansingh Stadium"       "St George's Park"      "SuperSport Park"       "Vidarbha Cricket Association Stadium, Jamtha"  "Wankhe
de Stadium

Enter the desired venue: MA Chidambaram Stadium, Chepauk
Choose pitch: 1.batting pitch , 2.balling pitch :
Enter the pitch : balling pitch

Batting Parameters Tuning begins...
Batting Prediction accuracy=0.4666666666666667 with classifier=SVC
Batting Prediction begins...
Batting Prediction Ends!

Bowling Parameter Tuning begins...
The bowling prediction accuracy=1.0 with classifier=XGB
Bowling Prediction begins...
Bowling Prediction Ends!


Number of predicted runs:0-10
Number of predicted wickets:0
```

**Fig.11 Output**

# Chapter VI

# Testing

| Test Case ID | Test Scenarios | Test Case | Preconditions | Test steps | Test Data | Expected results | postconditions | Actual Results | Status Pass /Fail |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Train | Batting | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed batting data | The number of runs scored. | The performance prediction model is generated based on training data | The number of runs scored. | Pass |
| 2 | Test | Batting | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed batting dataset | The number of runs scored. | The performance prediction model is generated based on testing data | The number of runs scored. | Pass |
| 3 | Train | Bowler | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed bowling dataset | The number of wickets taken | The performance prediction model is generated based on training data | The number of wickets taken | Pass |
| 4 | Test | Bowler | Against an opposition team in a desired venue at a particular | Load the data, extract the feature attributes, | Preprocessed bowling datase | The number of wickets taken | The performance prediction model is generated | The number of wickets taken | Pass |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | pitch | and predict the runs based on the data available. | t | | based on training data | | |
| 5 | Train | Allrounder | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed bowling and Batting dataset | The number of wickets taken and runs scored | The performance prediction model is generated based on training data | The number of wickets taken and runs scored | Pass |
| 6 | Test | Allrounder | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed bowling and Batting dataset | The number of wickets taken and runs scored | The performance prediction model is generated based on training data | The number of wickets taken and runs scored | Pass |
| 7 | Train | Batting | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed batting data | The number of runs scored | The performance prediction model is generated based on training data | The number of runs scored | Pass |
| 8 | Test | Batting | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and | Preprocessed batting data | The number of runs scored | The performance prediction model is generated based on | The number of runs scored | Pass |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | predict the runs based on the data available. | | | training data | | |
| 9 | Train | Bowler | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed bowling dataset | The number of wickets taken | The performance prediction model is generated based on training data | The number of wickets taken | Pass |
| 10 | Test | Bowler | Against an opposition team in a desired venue at a particular pitch | Load the data, extract the feature attributes, and predict the runs based on the data available. | Preprocessed bowling dataset | The number of wickets taken | The performance prediction model is generated based on training data | The number of wickets taken | Pass |

**Table.1 Test Cases**

# Chapter VII

# Conclusion& Future Scope

## Conclusion:

In summary, the main objective of this project was to precisely evaluate the performance of cricket players in terms of runs scored by batsmen, wickets taken by bowlers, and the overall performance of all-rounders in terms of runs and wickets. By employing various data analysis techniques, statistical models, and potentially machine learning algorithms, the project successfully assessed players' performance using historical data or real-time match information.

The achieved accuracy of 0.86 demonstrates a significant correlation between the model's predictions and the actual outcomes, indicating a high level of dependability in predicting cricket players' performance. This level of accuracy is valuable for coaches, selectors, and team management, enabling them to make informed decisions regarding player selection, team composition, and strategic planning.

By leveraging data-driven approaches, this project enhances the understanding of player performance and contributes to the advancement of cricket analysis. The precise evaluation of players' performance in different aspects of the game empowers teams to optimize their strategies and enhance overall team performance. This project serves as a foundation for further advancements in predicting player performance and contributes to the continuous evolution of the sport of cricket.

## Future Scope:

The cricket player performance prediction project holds promising potential for future advancements and improvements. Here are some areas that can be explored further:

**Advanced Machine Learning Techniques:** The project can explore the integration of advanced machine learning algorithms, such as deep learning models, ensemble methods, or reinforcement learning, to enhance the accuracy and reliability of player performance predictions. By incorporating these techniques, the project can provide more precise insights into player performance and enable better decision-making.

**Real-Time Data Analysis:** While the project currently relies on historical data and match information, there is room for incorporating real-time data analysis. By integrating live data feeds and implementing real-time analytics, the project can offer dynamic and up-to-date predictions during live matches. This enhancement would improve the timeliness and relevance of player performance predictions.

**Integration with Player Selection and Team Strategy:** The project's findings can be practically applied by integrating the performance predictions with player selection and team strategy decisions. By considering the predicted performance of players, teams can make informed choices about team composition, batting order, bowling strategies, and fielding placements, optimizing their chances of success.

# Chapter VIII

# References


1. An efficient team prediction for one day international matches using a hybrid approach of CS-PSO and machine learning algorithms
2. Sklearn RandomForestClassifier
3. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow book

# Chapter IX

## List of Figures:



## List of Tables: