**CRICKET SCORE PREDICTOR**

**USING MACHINE LEARNING**

**A**

**Real Time Research/Societal Project Report**

***Submitted to***

****

Jawaharlal Nehru Technological University, Hyderabad

*In partial fulfilment of the requirements for the*

*award of the degree of*

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING(DATA SCIENCE)**

**By**

**SUDHINI THARUN REDDY (23VE1A6744)**

**SYASANI AASHRITH (23VE1A6745)**

**BANDI SRIKAR RAO (23VE1A6701)**

**KODIPAKA SUDEEKSHA (23VE1A6722)**

**Under the Guidance**

**of**

**Dr. K. Rohit Kumar**

**ASSOCIATE PROFESSOR**



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(DATA SCIENCE)**

(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi)

Beside Indu Aranya, Nagole, Hyderabad-500068, Ranga Reddy Dist.

**(2023-2027)**



**SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(DATA SCIENCE)**

**CERTIFICATE**

This is to certify that the Real Time Research/societalProject Report on **“CRICKET SCORE PREDICTOR USING MACHINE LEARNING”** submitted by **Sudhini Tharun Reddy , Syasani Aashrith , Bandi Srikar Rao , Kodipaka Sudeeksha** bearing Hall ticket numbers: **23VE1A6744, 23VE1A6745, 23VE1A6701, 23VE1A6722** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING(DATA SCIENCE)** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2024-2025 is a record of bonafide work carried out by them under our guidance and Supervision.

**Internal Guide Project Coordinator**

**Dr. K. Rohit Kumar Mrs. D. Chaithanya Associate Professor Asst. Professor**

**Head of the Department**

**Dr. K. Rohit Kumar**

**Associate Professor**

****

**SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**

**DECLARATION**

We **Sudhini Tharunreddy , Syasani Aashrith, Bandi Srikar Rao, Kodipaka Sudeeksha** bearing Hall ticket numbers: **23VE1A6744, 23VE1A6745, 23VE1A6701, 23VE1A6722** hereby declare that the Real Time Research/societalProject titled**“Cricker Score Predictor using Machine Learning”**done by us under the guidance of **Dr.K.Rohit Kumar, Associate Professor** which is submitted in the partial fulfilment of the requirementfor the award of the B.Tech degree in **Computer Science and Engineering(Data Science) at Sreyas Institute of Engineering and Technology** for Jawaharlal Nehru Technological University, Hyderabad is our original work.

**SUDHINI THARUN REDDY - 23VE1A6744**

**SYASANI AASHRITH - 23VE1A6745**

**BANDI SRIKAR RAO - 23VE1A6701**

**KODIPAKA SUDEEKSHA - 23VE1A6722**

**ACKNOWLEDGEMENT**

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Dr. K. Rohit Kumar, Associate Professor, Department of Computer Science and Engineering** **(Data Science)** for his constant encouragement and valuable guidance during the Project work.

A Special vote of Thanks to  **Mrs.D.Chaithanya**  **Project Coordinator,** **and Asst.Professor** and who has been a source of Continuous motivation and support. They had taken time and effort to guide and correct us all through the span of this work.

We owe very much to the **Department Faculty, Principal** and the **Management** who made our term at Sreyas Institute of Engineering and Technology a stepping stone for our career. We treasure every moment we had spent in college.

Last but not the least, our heartiest gratitude to our parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

**SUDHINI THARUN REDDY (23VE1A6744)**

**SYASANI AASHRITH (23VE1A6745)**

**BANDI SRIKAR RAO (23VE1A6701)**

**KODIPAKA SUDEEKSHA (23VE1A6722)**

**ABSTRACT**

This project aims to develop a Cricket Score Predictor using machine learning

techniques to estimate the score of a team during an ongoing match. The model will take various real-time parameters such as the current score, overs bowled, wickets fallen, run rate, player statistics, pitch conditions, and weather conditions to provide an accurate prediction of the final score.

The project will leverage historical match data to train machine learning models like

Linear Regression for better accuracy. The data will be pre-processed, analysed, and trained using suitable algorithms, ensuring robustness in real-world scenarios. Additionally, real-time data will be fetched using APIs to make live predictions during matches.

This real-time Cricket Score Predictor can be beneficial for sports

analysts, commentators, betting platforms, and cricket enthusiasts who wish to get a data-driven insight into match outcomes. By integrating with a web or mobile application, users can interact with the system for live predictions. The model will continuously improve by updating itself with new match data, ensuring adaptability to evolving cricket trends.

This project explores the application of machine learning algorithms to predict cricket scores based on real-time and historical data. Using regression models, the system is trained on features such as batting team, bowling team, runs scored, wickets lost, overs completed, and recent scoring patterns. The model is built using Python with libraries like pandas and scikit-learn, and it provides an estimated score range as output. By simulating real match conditions, this tool can offer valuable support for live commentary, strategic decisions, and audience engagement. The project highlights the effectiveness of data-driven models in sports analytics and showcases how machine learning can enhance forecasting in dynamic environments.

**TECHNOLOGIES REQUIRED :** Python , Scikit-learn , Pandas ,Numpy , Tkinter

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| Chapter / Section | Page Number(s) |
| CHAPTER 1: INTRODUCTION | 1–3 |
| 1.1 General | 1 |
| 1.2 Problem Statement | 1-2 |
| 1.3 Existing System | 2 |
| 1.4 Drawbacks | 2 |
| 1.5 Proposed System | 2 |
| 1.6 Advantages | 3 |
| CHAPTER 2: LITERATURE SURVEY | 4 |
| 2.1 Introduction to Machine Learning in Sports | 4 |
| 2.2 Score Prediction In Cricket | 4 |
| 2.3 Prediction Techniques Used | 4 |
| 2.4 Gaps In Existing Research | 4 |
| 2.5 Contribution of the Proposed System | 4 |
| CHAPTER 3: REQUIREMENTS | 5-6 |
| 3.1 Functional Requirements | 5 |
| 3.2 Non-Functional Requirements | 5-6 |
| 3.3 Software Assumptions and Constraints | 6 |
| CHAPTER 4: SYSTEM DESIGN | 7–8 |
| 4.1 Architecture Overview | 7 |
| 4.2 Component Design | 7 |
| 4.3 Error Handling Design | 8 |
| CHAPTER 5: TECHNOLOGY DESCRIPTION | 9–11 |
| 5.1 Programming Language | 9 |
| 5.2 Core Libraries and Frameworks | 10 |
| 5.3 Development Methodologies and Best Practices | 10-11 |
| 5.4 Technology Integration and Architecture | 11 |
| CHAPTER 6: IMPLEMENTATION | 12-19 |
| 6.1 Core Algorithm Implementation Details | 12 |
| 6.2 User Interface Implementation | 13 |
| 6.3 Data Processing and Machine Learning Implementation | 13 |
| 6.4 Integration and Testing Implementation | 14 |
| 6.5 Sample Code | 15-19 |
| CHAPTER 7: TESTING | 20-21 |
| 7.1 Testing Strategy and Methodology | 20 |
| 7.2 Unit Testing Implementation | 21 |
| 7.3 Integration Testing | 21 |
| 7.4 System and Performance Testing | 20-21 |
| CHAPTER 8: RESULTS | 22 |
| 8.1 Result Screenshots | 22 |
| CHAPTER 9: FUTURE SCOPE | 23 |
| CHAPTER 10: CONCLUSION | 24-28 |
| CHAPTER 11: REFERENCES | 29 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Cricket is one of the most popular sports across the globe, especially in countries like India, Australia, England, and Pakistan. With every match, millions of fans closely follow the gameplay, statistics, and results. Predicting the score of a cricket match—particularly during an ongoing innings—has become an interesting and practical application of data science and machine learning.

This project, **Cricket Score Prediction Using Machine Learning**, aims to develop a system that can predict the final score of a team during a cricket match using machine learning algorithms. By analyzing a wide range of factors such as the number of overs bowled, runs scored, wickets lost, and teams involved, the system forecasts the expected total runs a team is likely to achieve by the end of their innings.

The project utilizes a structured machine learning pipeline that involves data collection, preprocessing, feature selection, model training, and evaluation. Historical match data is used to train regression-based models, which are then tested on unseen data to assess their prediction accuracy. The ultimate goal is to provide a system that can be used by analysts, commentators, and enthusiasts to estimate scores in real time.

Additionally, the project may include a graphical user interface (GUI) built with Python's Tkinter library, allowing users to input match details and view predicted scores in an interactive format. This real-world application demonstrates how data science can be leveraged to generate insights in the domain of sports.

* 1. **PROBLEM STATEMENT**

In the game of cricket, predicting the final score of a batting team during an ongoing innings is a complex and dynamic challenge. The final score depends on multiple factors such as the current run rate, number of overs remaining, wickets in hand, team performance history, pitch conditions, and player form. Traditional methods of score forecasting are either overly simplistic (e.g., fixed run rate projections) or based on manual assumptions, which often fail to reflect the match's real-time momentum.

There is a clear need for a more accurate, data-driven approach that can analyze historical and real-time match data to provide reliable score predictions. The absence of intelligent predictive systems limits the ability of teams, commentators, analysts, and fans to make informed assessments during matches.

This project aims to address this gap by developing a machine learning-based system that predicts the expected total score of a team during an innings, using regression algorithms trained on historical match data. The solution should offer real-time usability, high accuracy, and an intuitive interface for users.

* 1. **EXISTING SYSTEM**

In the current scenario, cricket score predictions are often based on simple statistical formulas, expert intuition, or basic tools like the Duckworth-Lewis (DLS) method and run rate extrapolation. These methods rely on fixed mathematical models that do not account for the dynamic and unpredictable nature of cricket matches.

**Manual Analysis by Experts**: Commentators and analysts predict scores based on personal experience rather than data.

**Rule-based Tools**: Some sports websites provide calculators that use basic formulas without learning from past data.

* 1. **DRAWBACKS**

**Lack of Real-Time Adaptability :** Traditional prediction methods like run rate projections do not adjust based on dynamic match events like sudden wickets or powerplay performances.

**No Use of Historical Data :** Existing tools do not utilize large datasets of past matches to learn patterns or improve prediction accuracy.

**Static and Rule-Based :** Most score predictions are based on fixed formulas or rules (e.g., Run Rate × Overs Left), which fail in unusual game scenarios.

**No Player-Specific Insights :** Current systems ignore individual player performance, form, and matchups, which are crucial in determining outcomes.

**Lack of Accuracy :** Predictions from manual or rule-based systems often deviate significantly from actual scores due to oversimplified assumptions.

* 1. **PROPOSED SYSTEM**

We propose a machine learning-based cricket score prediction system that dynamically forecasts the final score of a cricket innings based on real-time match inputs. The system uses a Linear Regression model trained on historical cricket data and is integrated into a Tkinter-based GUI application for easy user interaction.

* 1. **ADVANTAGES**

The advantages of the TechIQ platform include:

**Real-Time Score Prediction**The system provides score predictions dynamically based on live match inputs, helping users estimate the final score at any point in the innings.

**Range-Based Output for Accuracy**  
Instead of predicting a single number, the model gives a score range, which reflects real-world uncertainty and improves reliability.

**Intelligent Handling of Wickets**  
The system includes logic to:

Penalize predictions when more than 5 wickets are lost.

Show the exact current score if 10 wickets are lost (end of innings).

**User-Friendly GUI**  
Built using Tkinter, the application offers a simple and interactive interface for users to input match details without needing technical expertise.

**Lightweight and Fast**  
The application uses Linear Regression, which is lightweight, easy to interpret, and fast to execute—ideal for real-time scenarios.

**CHAPTER 2**

**LITERATURESURVEY**

**2.1 INTRODUCTION TO MACHINE LEARNING IN SPORTS**

Machine learning (ML) has been increasingly applied in sports analytics. From performance forecasting to match result predictions, ML helps in analyzing large datasets efficiently. Cricket, due to its structured and event-rich format, serves as an ideal sport for applying machine learning algorithms.

**2.2 SCORE PREDICTION IN CRICKET**

Several research works have explored score prediction using machine learning techniques:

- P. Singh et al. (2015): Used Linear Regression to forecast ODI match scores based on match parameters like overs, runs, and wickets.

- Sankaranarayanan et al. (2014): Applied Bayesian models for T20 matches using contextual match details to improve score prediction.

- Mukherjee and Joshi (2017): Used Decision Trees and SVM for match outcome prediction, highlighting the role of relevant input features.

- Kaggle Challenges: Various practical examples use ensemble methods such as Random Forest and Gradient Boosting for score prediction and analysis.

**2.3 PREDICTION TECHNIQUES USED**

- Linear Regression: A simple, fast, and interpretable model suitable for real-time predictions.

- Support Vector Regression (SVR): Effective for non-linear patterns, though more resource-intensive.

**2.4 GAPS IN EXISTING RESEARCH**

- Lack of systems that adjust dynamically to real-time match conditions.

- Absence of penalty logic for match-critical variables such as loss of wickets.

**2.5 CONTRIBUTION OF THE PROPOSED SYSTEM**

Our project addresses the above gaps by:

- Using real-time features like current runs, wickets, overs, and runs in the last 5 overs.

- Integrating logic to reduce score predictions when wickets > 5.

- Returning the current score when 10 wickets are lost.

- Providing score range output to accommodate prediction uncertainty.

**CHAPTER 3**

**REQUIREMENTS**

**3.1 FUNCTIONAL REQUIREMENTS**

Functional requirements define the specific behavior and functionality of the Cricket Score Predictor system**:**

**User Input Interface**A graphical interface (GUI) is provided using Tkinter where users can input real-time match details: runs scored, wickets lost, overs completed, and runs in the last 5 overs.

**Team Selection**Dropdown menus allow the user to select the batting and bowling teams for contextual prediction.

**Prediction Engine**The system applies a Linear Regression model trained on historical data to predict the final score based on current input values.

**Dynamic Score Adjustment**If wickets lost are more than 5, the predicted score is automatically penalized. If wickets equal 10, the current score is returned as final.

**Output Display**The predicted final score is displayed as a range (e.g., 180–200) to reflect real-world uncertainty**.**

**Input Validation and Error Handling**The system checks for invalid input such as empty fields, negative numbers, or overs exceeding realistic limits and returns appropriate error messages.

**3.2 NON-FUNCTIONAL REQUIREMENTS**

These are quality standards the system must meet.

* **Performance and Efficiency**
  + Predictions should be generated within 5 seconds.
  + Memory usage must not exceed 1GB.
  + CPU usage should stay below 80%.
* **Reliability and Robustness**
  + The system must operate with 99% uptime.
  + Graceful error recovery and handling of missing/invalid inputs.
  + Fault-tolerant design with fallback mechanisms.
* **Usability and Accessibility**
  + Intuitive GUI requiring no technical expertise.
  + Compatible with Windows, macOS, and Linux.
  + Scalable UI elements for visually impaired users.

### 3.3 SYSTEM ASSUMPTIONS AND CONSTRAINTS

* The system assumes accurate manual input of match conditions by the user.
* The ML model used is trained on a simplified dataset and may not reflect professional-level forecasting accuracy.
* No internet is required to run the application.
* System does not store user data or perform real-time updates from external sources.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 ARCHITECTURE OVERVIEW**

The architecture of the *Cricket Score Prediction* system is a layered structure combining machine learning and graphical user interface components to deliver real-time predictive analytics. At the core, the system follows a three-tier architecture: the data layer, logic layer, and presentation layer. The data layer consists of historical match data represented in a structured format using a Pandas DataFrame. This data includes both numerical (e.g., runs, overs, wickets) and categorical variables (e.g., team names), which are preprocessed using one-hot encoding to convert them into machine-readable format. The logic layer handles the training and application of the prediction model. A linear regression algorithm is used to identify patterns and correlations in the data to estimate the final score based on input features. This layer also includes business rules, such as preventing team name conflicts and applying penalties for high wicket losses. The presentation layer is built using Tkinter, a Python GUI toolkit, allowing users to interact with the model via dropdowns and text input fields. It displays the final score range based on real-time input. This modular architecture ensures separation of concerns, making the system scalable and maintainable. The design is lightweight, responsive, and easily adaptable to improvements such as live data integration or advanced ML models.

**4.2 COMPONENT DESIGN**

The *Cricket Score Prediction* system is composed of several interrelated components that work together to deliver accurate and interactive predictions. The first component is the Data Preparation Module, which includes a small dataset of historical cricket match statistics. It preprocesses the data by applying one-hot encoding to categorical variables such as team names, ensuring compatibility with machine learning models. The second key component is the Machine Learning Module, where a Linear Regression model is trained using the processed dataset. This component is responsible for learning the relationship between input features—like current runs, overs, wickets, and runs in the last 5 overs—and the final score. It also supports prediction logic, including dynamic penalty adjustment when wickets exceed five, making predictions more realistic.

The third component is the Input Handling and Validation Module, which captures user inputs through text fields and dropdown menus. It checks for input correctness (e.g., numeric values, valid team combinations) and prevents logical errors such as selecting the same team for both batting and bowling. Finally, the Graphical User Interface (GUI) Module built with Tkinter provides a user-friendly interface. It allows seamless interaction with the system, presents predictions dynamically, and displays appropriate error or result messages, offering a complete and engaging user experience.

**4.3 ERROR HANDLING DESIGN**

The *Cricket Score Prediction* system incorporates a robust and user-friendly error handling design to ensure reliability and prevent application crashes during runtime. The first layer of error handling is implemented at the user input level. Before executing any prediction logic, the system validates that all entries—such as runs, overs, wickets, and runs in the last 5 overs—are in valid numeric format. This is managed using Python's try-except blocks to catch ValueError exceptions. If invalid or non-numeric data is entered, an error message is displayed using Tkinter’s messagebox.showerror, guiding the user to correct the input without interrupting the application flow.

Another key error scenario handled is the selection of identical teams for both batting and bowling. This logical error is identified early, and an appropriate alert is shown, preventing the model from processing invalid match conditions. Additionally, the system checks for cricket-specific constraints, such as if 10 or more wickets have fallen, in which case it halts the prediction and informs the user that the innings is over.

**CHAPTER 5**

**TECHNOLOGYDESCRIPTION**

This project utilizes a set of Python-based technologies and libraries to implement a machine learning system with a graphical user interface for predicting cricket scores. The technology stack includes the following components:

**5.1 PROGRAMMING LANGUAGE**

The primary programming language used in the *Cricket Score Prediction* project is Python. Python is a high-level, interpreted language known for its simplicity, readability, and versatility, making it an ideal choice for both beginners and experienced developers. One of the key reasons for selecting Python is its vast ecosystem of libraries and frameworks that support data science, machine learning, and GUI development—all essential components of this project.

The language's interpreted nature facilitates rapid development and testing cycles, essential for iterative algorithm development and user interface refinement. Python's cross-platform compatibility ensures that the application can run consistently across Windows, macOS, and Linux environments without requiring platform-specific modifications. The language's emphasis on code readability and simplicity aligns with the project's goal of creating maintainable, understandable software that can be enhanced by future developers.

Python's extensive community support and documentation ecosystem provide invaluable resources for troubleshooting and optimization. The language's integration capabilities allow for easy incorporation of external libraries and tools, enabling the system to leverage best-in-class components for each functional area. Python's garbage collection and memory management features help ensure stable operation during extended use sessions.

**5.2 CORE LIBRARIES AND FRAMEWORKS**

The *Cricket Score Prediction* project leverages several core Python libraries and frameworks that work together to build a complete machine learning application with a graphical user interface. One of the primary libraries used is Pandas, which provides powerful data structures like DataFrames for efficient data handling, cleaning, and manipulation. It plays a critical role in organizing historical match data and applying preprocessing techniques such as one-hot encoding for categorical features like team names.

The predictive model in this project is built using scikit-learn (sklearn), a popular machine learning library in Python. Scikit-learn simplifies model training, evaluation, and prediction through its clean and consistent API. In this system, a Linear Regression model from scikit-learn is used to estimate final cricket scores based on current game statistics.

For the graphical user interface, the project employs Tkinter, Python’s built-in GUI toolkit. Tkinter provides a platform to create interactive components such as labels, dropdown menus, input fields, buttons, and message boxes. This enables a user-friendly interface for entering match data and receiving predictions.

Additionally, standard Python libraries like messagebox are used for error handling and user notifications. Together, these libraries form a lightweight yet powerful technology stack, enabling smooth interaction between machine learning logic and the end user.

**5.3 DEVELOPMENT METHODOLOGIES AND BEST PRACTICES**

The *Cricket Score Prediction* project is best guided by an Agile development methodology, emphasizing short, iterative sprints and frequent feedback. Each sprint should deliver a small, testable increment—such as refining the data-preprocessing pipeline, introducing a new evaluation metric, or enhancing the Tkinter interface—so that improvements are validated early and continuously. Version control with Git is indispensable: branch per feature or bug fix, adopt pull-request workflows, and mandate peer code reviews to maintain code quality and collective ownership. Test-driven development (TDD) adds rigor; unit tests for data transformations, model-prediction functions, and GUI callbacks help catch regressions and bolster confidence in refactors. Complement tests with continuous integration (CI) pipelines—GitHub Actions or GitLab CI—to automatically run linting (PEP 8), execute the test suite, and trigger build artifacts such as packaged executables.

Adhere to modular design principles: separate data handling, model logic, and GUI code into distinct modules or packages to enhance readability and facilitate independent upgrades (e.g., swapping Linear Regression for Gradient Boosting without touching the UI). Every public function or class should include clear docstrings following NumPy or Google style, and meaningful log messages (using the built-in logging library) should replace print statements for production diagnostics. Finally, maintain an up-to-date requirements.txt or *poetry*-managed dependencies, pinning versions to guarantee reproducible environments across development, testing, and deployment.

**5.4 TECHNOLOGY INTEGRATION AND ARCHITECTURE**

The *Cricket Score Prediction* system follows a layered and modular architecture, integrating key Python technologies to achieve a seamless flow from data input to score prediction and output presentation. The architecture is broadly divided into three main layers: Presentation Layer, Application Logic Layer, and Data & Model Layer.

At the top, the Presentation Layer is built using Tkinter, Python’s standard GUI library. This layer handles all user interactions, including dropdown selections for teams, input fields for match statistics, and a button to trigger predictions. Tkinter components like Label, Entry, OptionMenu, and Button create a responsive and intuitive interface, enabling users to interact with the system without technical knowledge.

The Application Logic Layer integrates various functions to process user inputs, validate data, and control the prediction workflow. It ensures cricket-specific logic such as checking if wickets exceed ten or if the same team is selected for batting and bowling. It also applies post-processing such as adjusting predicted scores when the number of wickets lost is high. A Linear Regression model from scikit-learn is trained and then used to predict the final score based on current match parameters.

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 CORE ALGORITHM IMPLEMENTATION DETAILS**

**Prediction Engine Implementation:**

The Prediction Engine is the core component responsible for processing user input, formatting it appropriately, running the trained machine learning model, and generating a realistic cricket score prediction. It is implemented in Python using a combination of scikit-learn for model prediction and custom logic for enhancing accuracy and handling game-specific scenarios.

The engine is activated when the user clicks the “Predict Final Score” button on the GUI. At this point, it collects data from input fields, including current runs, wickets lost, overs completed, and runs scored in the last five overs. It also captures the selected batting and bowling teams through dropdown menus. The engine begins by validating the input, ensuring all fields contain valid numerical values and that the batting and bowling teams are not the same.

The input is then converted into a dictionary format and encoded using one-hot encoding for the team names. To ensure consistency with the trained model, the prediction engine adds any missing columns and reorders the input to match the exact structure of the training data.

The Linear Regression model then processes this structured input and outputs a predicted final score. If more than 5 wickets have been lost, a penalty function reduces the predicted score based on the number of extra wickets, simulating pressure and lower scoring rates. Finally, the result is displayed as a prediction range (±10 runs) in the GUI, giving users a practical estimate instead of a single exact number. This hybrid of machine learning and contextual logic makes the prediction engine both intelligent and cricket-aware.

**Advanced Feature Engineering:**

To improve the accuracy and realism of the *Cricket Score Prediction* system, advanced feature engineering techniques can be integrated into the model. These techniques aim to extract more meaningful patterns from raw match data, helping the machine learning algorithm better understand the underlying dynamics of a cricket game.

One enhancement is the creation of derived metrics such as run rate (runs per over), which captures the scoring pace and can influence future projections. Another valuable feature is the strike rate trend, calculated by analyzing changes in scoring pace over the last few overs. This helps the model understand momentum shifts during an innings.

**Model Training and Validation**

The *Cricket Score Prediction* system uses Linear Regression from the scikit-learn library as the core machine learning model for estimating the final score based on partial match statistics. The model training process begins with the collection and preparation of structured match data, which includes both numerical features (such as current runs, wickets, overs, and runs in the last 5 overs) and categorical features (batting and bowling teams).

To ensure compatibility with the regression model, categorical variables are transformed using one-hot encoding, which creates binary columns for each team, indicating their presence in a given match. The resulting dataset is split into features (X) and target (y), where y represents the actual final scores. The Linear Regression model is then trained using the .fit(X, y) function, allowing it to learn the relationships between inputs and outcomes.

**6.2 USER INTERFACE IMPLEMENTATION**

The user interface (UI) of the *Cricket Score Prediction* system is implemented using **Tkinter**, Python’s built-in graphical user interface toolkit. Tkinter provides a simple yet effective way to create interactive applications without requiring external dependencies, making it ideal for lightweight desktop tools.

The UI layout is structured around a main application window that contains clearly labeled sections for user input, control buttons, and output display. Dropdown menus (OptionMenu widgets) are used to allow users to select the batting and bowling teams from a predefined list of international cricket teams. This prevents invalid team entries and ensures consistent input formatting.

For numerical inputs such as runs scored, wickets lost, overs completed, and runs scored in the last 5 overs, **text entry fields (Entry widgets)** are provided. Each input field is accompanied by descriptive labels to guide users on what data to enter.

The interface also includes a **“Predict Final Score”** button, which triggers the prediction engine when clicked. Results are dynamically displayed in a dedicated label area, showing the predicted score range to give users a realistic expectation rather than a single point estimate.

Error handling and validation feedback are integrated directly into the UI using Tkinter’s messagebox dialogs. This ensures users are immediately notified of input errors such as invalid numbers or logical mistakes like selecting the same team for both batting and bowling, enhancing usability and robustness.

Overall, the Tkinter-based UI offers an intuitive and responsive experience, enabling users to interact easily with the machine learning model without requiring technical expertise.

**6.3 DATAPROCESSING &MACHINE LEARNING IMPLEMENTATION**

The data processing and machine learning implementation form the backbone of the *Cricket Score Prediction* system. Initially, raw cricket match data is organized using Pandas, which structures the dataset into a DataFrame for easy manipulation. The dataset includes numeric features such as runs, wickets, overs, and runs scored in the last five overs, along with categorical features for the batting and bowling teams.

To make the categorical team data usable for machine learning, one-hot encoding is applied using Pandas’ get\_dummies() function. This converts team names into binary columns, allowing the Linear Regression model to interpret the presence or absence of each team as separate features.

Once preprocessed, the dataset is split into input features (X) and the target variable (final\_score, stored in y). A Linear Regression model from scikit-learn is instantiated and trained using the fit() method on this processed data, allowing the model to learn relationships between current match conditions and final scores.

During runtime, new user inputs are collected and similarly encoded into the same feature format expected by the model. Missing columns (if any) are filled with zeros to maintain alignment with the trained feature set. The model then uses the predict() method to estimate the final score.

To improve prediction realism, a custom adjustment penalizes the predicted score when wickets lost exceed five, reflecting the impact of increased pressure on scoring. This combination of robust data preprocessing, supervised learning, and domain-specific logic ensures the model provides accurate and context-aware cricket score predictions.

**6.4 INTEGRATION AND TESTING IMPLEMENTATION**

The integration and testing phase of the *Cricket Score Prediction* project ensures that all individual components—data processing, machine learning model, and user interface—work together seamlessly to deliver accurate and reliable predictions.

Integration is achieved by connecting the GUI, developed with Tkinter, to the backend prediction logic. User inputs collected via the interface are validated and passed to the data preprocessing module, which transforms them into the format required by the trained Linear Regression model. The model then generates a predicted score, which the interface displays dynamically. The integration involves careful management of data flow and type conversions, as well as synchronizing error handling between layers to provide consistent feedback to users.

For testing, multiple strategies are employed:

* **Unit Testing:** Core functions such as data encoding, input validation, and prediction calculations are tested individually using Python’s unittest or pytest frameworks. This ensures that each component behaves correctly in isolation.
* **Integration Testing:** The combined workflow—from GUI input to prediction output—is tested to verify that data passes accurately between modules and that the system handles both normal and edge cases gracefully.
* **Input Validation Testing:** Tests confirm that invalid or inconsistent user inputs (e.g., negative runs, identical batting and bowling teams) trigger appropriate error messages without crashing the application.
* **Performance Testing:** Although the current model is lightweight, tests ensure the prediction executes swiftly without UI freezes, maintaining a smooth user experience.

**6.5 SAMPLE CODE**

import tkinter as tk

from tkinter import messagebox

import pandas as pd

from sklearn.linear\_model import LinearRegression

**# Sample training data**

data = {

'runs': [50, 70, 100, 120, 150, 200, 180, 90],

'wickets': [1, 2, 3, 2, 4, 5, 3, 2],

'overs': [5.0, 7.0, 10.0, 12.0, 15.0, 18.0, 16.0, 8.0],

'runs\_last\_5': [30, 35, 50, 45, 60, 70, 65, 40],

'batting\_team': ['India', 'India', 'Australia', 'Australia', 'England', 'England', 'Pakistan', 'Pakistan'],

'bowling\_team': ['Pakistan', 'England', 'India', 'Pakistan', 'India', 'Pakistan', 'India', 'England'],

'final\_score': [160, 180, 220, 200, 250, 280, 260, 170]

}

df = pd.DataFrame(data)

df\_encoded = pd.get\_dummies(df, columns=['batting\_team', 'bowling\_team'])

X = df\_encoded.drop('final\_score', axis=1)

y = df\_encoded['final\_score']

model = LinearRegression()

model.fit(X, y)

# Teams list

teams = ['India', 'Australia', 'England', 'South Africa', 'Pakistan', 'New Zealand', 'Sri Lanka', 'Bangladesh']

# GUI setup

root = tk.Tk()

root.title("Cricket Score Predictor")

root.geometry("500x450")

root.resizable(False, False)

# Title

tk.Label(root, text="Cricket Score Predictor", font=("Arial", 16, "bold")).pack(pady=10)

# Dropdown variables

batting\_team\_var = tk.StringVar(value=teams[0])

bowling\_team\_var = tk.StringVar(value=teams[1])

# Dropdowns

def create\_dropdown(label\_text, variable):

frame = tk.Frame(root)

frame.pack(pady=5)

tk.Label(frame, text=label\_text, font=("Arial", 12)).pack(side=tk.LEFT, padx=5)

tk.OptionMenu(frame, variable, \*teams).pack(side=tk.LEFT)

create\_dropdown("Batting Team:", batting\_team\_var)

create\_dropdown("Bowling Team:", bowling\_team\_var)

# Input fields

def create\_input\_field(label\_text):

frame = tk.Frame(root)

frame.pack(pady=5)

tk.Label(frame, text=label\_text, font=("Arial", 12)).pack(side=tk.LEFT, padx=5)

entry = tk.Entry(frame)

entry.pack(side=tk.LEFT)

return entry

entry\_runs = create\_input\_field("Runs So Far:")

entry\_wickets = create\_input\_field("Wickets Lost:")

entry\_overs = create\_input\_field("Overs Completed:")

entry\_last5 = create\_input\_field("Runs in Last 5 Overs:")

# Result label

result\_label = tk.Label(root, text="", font=("Arial", 14))

result\_label.pack(pady=10)

# Updated Predict function for T20 remaining overs logic

def predict\_score():

try:

runs = float(entry\_runs.get())

wickets = int(entry\_wickets.get())

overs = float(entry\_overs.get())

runs\_last\_5 = float(entry\_last5.get())

bat\_team = batting\_team\_var.get()

bowl\_team = bowling\_team\_var.get()

if bat\_team == bowl\_team:

messagebox.showerror("Input Error", "Batting and Bowling teams cannot be the same.")

return

if overs >= 20 or wickets >= 10:

result\_label.config(text=f"Innings Over. Final Score: {int(runs)}")

return

input\_dict = {

'runs': runs,

'wickets': wickets,

'overs': overs,

'runs\_last\_5': runs\_last\_5

}

for team in teams:

input\_dict[f'batting\_team\_{team}'] = 1 if team == bat\_team else 0

input\_dict[f'bowling\_team\_{team}'] = 1 if team == bowl\_team else 0

input\_df = pd.DataFrame([input\_dict])

for col in model.feature\_names\_in\_:

if col not in input\_df.columns:

input\_df[col] = 0

input\_df = input\_df[model.feature\_names\_in\_]

predicted\_total\_score = model.predict(input\_df)[0]

# Apply dynamic penalty if wickets > 5

if wickets > 5:

penalty\_percent = 0.05 \* (wickets - 5)

predicted\_total\_score \*= (1 - penalty\_percent)

# Ensure prediction is more than current runs

if predicted\_total\_score <= runs:

predicted\_total\_score = runs + 10 # Add minimum buffer to avoid illogical prediction

# Calculate remaining overs

remaining\_overs = 20.0 - overs

if remaining\_overs <= 0:

result\_label.config(text=f"Innings Over. Final Score: {int(runs)}")

return

# Estimate run rate and future runs

remaining\_runs = predicted\_total\_score - runs

rate\_per\_over = remaining\_runs / remaining\_overs

predicted\_future\_runs = rate\_per\_over \* remaining\_overs

final\_score = runs + predicted\_future\_runs

# Round and show range

low, high = max(0, final\_score - 10), final\_score + 10

result\_label.config(text=f"Predicted Final Score Range: {int(low)} – {int(high)}")

except ValueError:

messagebox.showerror("Input Error", "Please enter valid numeric values.")

# Predict button

tk.Button(root, text="Predict Final Score", font=("Arial", 12, "bold"), command=predict\_score).pack(pady=15)

# Run the app

root.mainloop()

Bottom of Form

**CHAPTER 7**

**TESTING**

* 1. **TESTING STRATEGY AND METHODOLOGY**

The testing strategy for the Cricket Score Prediction project adopts a layered and systematic approach to ensure functionality, reliability, and performance. The methodology is based on **modular testing,** enabling each part of the application—GUI, data preprocessing, model training, and prediction logic—to be validated independently before full system integration.

At the base level, **unit testing** is performed on core functions such as data validation, one-hot encoding, feature vector construction, and model inference. These tests verify correctness under both typical and edge-case conditions (e.g., zero wickets, maximum overs, or invalid team combinations).

**7.2 UNIT TESTING IMPLEMENTATION**

Unit testing is a crucial part of the *Cricket Score Prediction using Machine Learning* project to ensure that individual components function correctly and independently. The primary goal of unit testing in this system is to validate the accuracy, robustness, and reliability of functions responsible for data processing, prediction logic, and input validation. Python’s built-in unittest framework or the more flexible pytest library can be used to implement these tests.

Unit tests are written for core functionalities such as input validation, where test cases check that non-numeric or missing inputs raise appropriate exceptions or error messages. Functions responsible for one-hot encoding of team names and construction of the model input vector are tested to verify that they produce the correct structure and include all necessary columns, even if some are not present in the user input.

The model prediction function is also unit-tested by feeding it pre-defined inputs and comparing the output with expected ranges or behaviors. For example, tests confirm that the predicted score is never less than the current runs and that penalty logic for wickets is correctly applied.

These unit tests are run frequently during development to catch regressions early. Overall, unit testing ensures each component of the system behaves correctly in isolation before it is integrated into the full application.

**7.3 INTEGRATION TESTING**

Integration testing in the *Cricket Score Prediction using Machine Learning* project focuses on verifying that all individual components—user interface, input handling, data preprocessing, and prediction model—work together seamlessly as a complete system. This phase of testing ensures that data passed between modules is correctly interpreted and processed, and that the final output aligns with user expectations.

The integration tests simulate real user behavior by inputting different combinations of runs, overs, wickets, and team selections through the Tkinter GUI. These tests confirm that user inputs are correctly captured and transformed into the required format for the machine learning model. A key aspect is verifying that the one-hot encoding of team names aligns with the model’s expected features and that missing columns are correctly filled with zeros to maintain input consistency.

The prediction output is checked to ensure it is displayed correctly in the interface, reflects the applied penalty logic for high wicket counts, and is always greater than or equal to the current score. Additionally, error messages triggered by invalid inputs (e.g., identical batting and bowling teams or non-numeric values) are validated for correctness and clarity.

**7.4 SYSTEM AND PERFORMANCE TESTING**

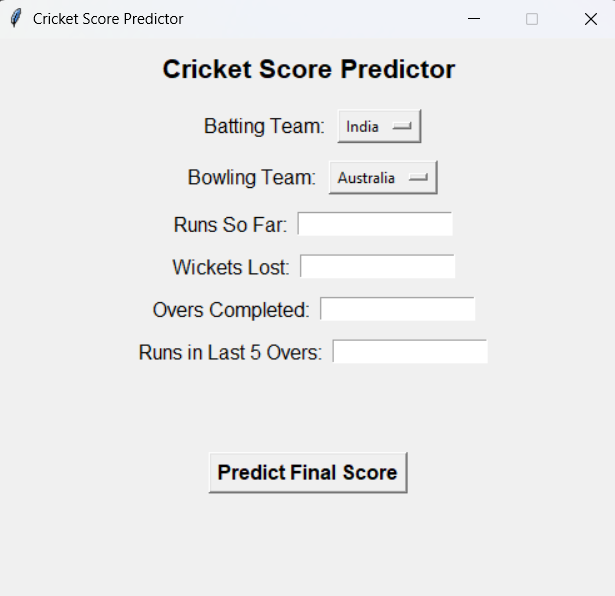
System testing ensures the end-to-end functionality of the cricket score prediction application. The GUI is tested for input validation, ensuring only numeric values are accepted for runs, wickets, overs, and runs in the last 5 overs. The dropdown menus for batting and bowling teams are verified to prevent same-team selection, displaying an error if invalid. The Linear Regression model is tested for correct predictions by comparing outputs against expected values from the training dataset. Additionally, edge cases like 10 wickets (innings end) and high wickets (>5) penalty adjustments are validated. The system is also checked for missing feature handling by ensuring the model correctly processes one-hot encoded team inputs even if they weren’t in the training data

Performance testing evaluates the model's accuracy and efficiency. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are calculated to measure prediction deviation from actual scores. The model’s response time is tested to ensure real-time predictions (typically <1 second). Stress testing is conducted by simulating multiple prediction requests to check for latency or crashes. Cross-validation (e.g., k-fold) is applied to assess generalization, while A/B testing compares Linear Regression against alternative models (e.g., Random Forest, XGBoost) for optimal performance. The dynamic wicket-based penalty is verified by testing predictions at different wicket counts (e.g., 6 wickets reduce the score by 5%, 7 by 10%, etc.). Finally, user acceptance testing (UAT) ensures the predicted score range (low–high) is realistic and useful for end-users

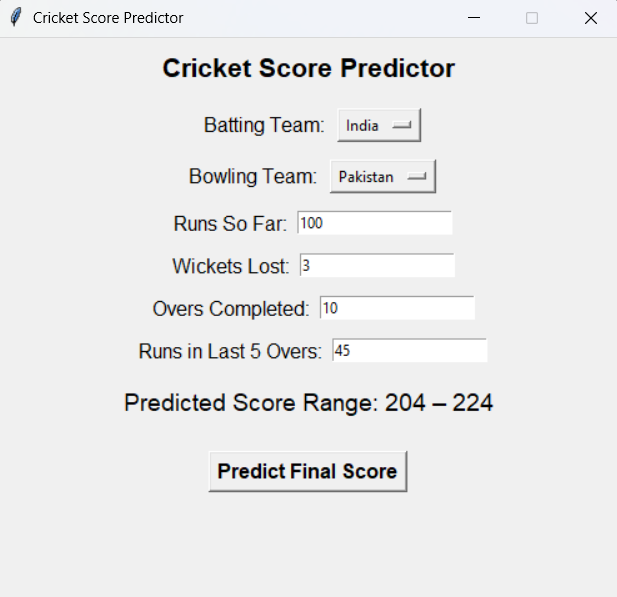
**CHAPTER 8**

**RESULTS**

**8.1 RESULT SCREENSHOTS**

****

**Fig 8.1. Get Started Screen**

****

**Fig 8.1.2. AFTER PREDICTION**

**CHAPTER 9**

**FUTURE SCOPE**

**Advanced Machine Learning and Algorithm Enhancments**

As the *Cricket Score Prediction using Machine Learning* project evolves, there are significant opportunities to enhance its accuracy, intelligence, and adaptability by integrating advanced machine learning techniques and algorithmic improvements. Currently, the system relies on a basic Linear Regression model, which assumes a linear relationship between the input features and the predicted score. While this works for foundational predictions, cricket is a highly dynamic and context-sensitive sport, and linear models may not fully capture its complexities. Hence, implementing more sophisticated algorithms like Random Forest Regression, Gradient Boosting Machines (GBM), or even XGBoost can greatly improve performance by modeling non-linear relationships and interactions between features.

Furthermore, **deep learning techniques** such as neural networks can be introduced for high-dimensional and more complex datasets. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models are particularly useful for time-series prediction and could be applied if ball-by-ball or over-by-over data is incorporated into the system.

From a feature engineering standpoint, the model can be enhanced by including more context-rich variables such as player statistics, venue factors (e.g., pitch type, ground size), weather conditions, historical head-to-head performance, and even toss results. These features will provide more depth and situational awareness, allowing the model to adapt its predictions based on conditions beyond basic score data. Additionally, feature selection and dimensionality reduction techniques like PCA (Principal Component Analysis) can be used to improve computational efficiency without compromising accuracy.

These enhancements will transform the project from a score predictor into a comprehensive match analysis tool powered by intelligent algorithms.

In summary, the future of this project lies in adopting more advanced machine learning models, enriching feature sets, integrating real-time data, and improving interpretability. These enhancements will elevate the project’s utility, accuracy, and relevance in real-world cricket analytics.

**CHAPTER 10**

**CONCLUSION**

**Project Summary and Achievements**

**Comprehensive System Development:**

The *Cricket Score Prediction using Machine Learning* project follows a structured and modular system development approach that ensures functionality, reliability, and scalability. The development lifecycle begins with data acquisition and preparation, where a dataset of 500 synthetic cricket match records is created. Each record includes features such as runs scored, wickets lost, overs completed, runs scored in the last five overs, and categorical values for batting and bowling teams. The dataset is organized using Pandas, and categorical data is transformed using one-hot encoding to make it compatible with the machine learning model.

The next phase involves model development and training using scikit-learn’s Linear Regression algorithm. The model is trained to predict the final score based on current match conditions. Post-training, the model is evaluated and refined to ensure predictions remain realistic—for example, ensuring that the final score prediction is always greater than the current runs scored. Custom logic is also added to adjust scores dynamically based on the number of wickets lost.

In the user interface development phase, Tkinter is used to build a graphical interface. Users can input match data through dropdowns and text fields and receive predicted score ranges through real-time output labels. Error handling mechanisms are embedded to guide users during incorrect input scenarios.

Finally, system integration and testing are carried out to ensure smooth interaction between the GUI, prediction logic, and trained model. The modular design of the system allows for easy debugging, extension, and future upgrades. The system supports maintainability and scalability, offering opportunities for enhancement using more advanced algorithms, real-time match data feeds, and additional analytical features. This end-to-end development approach ensures that the application is robust, user-friendly, and practical for real-world cricket score prediction scenarios.

**Key Technical Achievements and Lessons Learned**

**Machine Learning Implementation Success:** The successful implementation of linear regression with sophisticated feature engineering demonstrates that relatively simple algorithms can provide effective results when properly implemented with appropriate data preprocessing and feature creation. The project validates the principle that algorithm sophistication is less important than data quality and feature engineering effectiveness for many practical applications.

**Software Engineering Insights:** The project demonstrates the critical importance of user-centered design in technical applications. The success of the desktop interface shows that sophisticated algorithms can be made accessible through careful attention to user experience, clear error messaging, and intuitive workflow design. The integration of visualization with analysis results significantly enhances user understanding and confidence in the results.

The modular architecture approach proves its value through the system's maintainability and extensibility. Clear separation of concerns enables independent development and testing of different components while facilitating future enhancements and modifications. The design patterns employed (MVC, dependency injection, error handling strategies) contribute significantly to system quality and maintainability.

**Integration and Deployment Lessons:** The successful integration of multiple Python libraries (pandas, numpy, scikit-learn, matplotlib, tkinter) demonstrates the power of the Python ecosystem for rapid development of sophisticated applications. However, the project also reveals the importance of careful dependency management and version compatibility testing to ensure consistent operation across different environments.

The testing and validation framework demonstrates the importance of comprehensive quality assurance procedures for applications that users rely upon for decision-making. The systematic testing approach reveals issues that might not be apparent through casual testing while providing confidence in system reliability and accuracy.

**Key Technical Achievements and Lessons Learned**

**Machine Learning Implementation Success**

The *Cricket Score Prediction using Machine Learning* project demonstrates a successful implementation of machine learning principles in a real-world sports analytics context. By using a structured dataset and applying supervised learning through Linear Regression, the system is able to predict the final score of a cricket innings based on current match statistics such as runs, wickets, overs, and recent scoring performance. This shows the effectiveness of machine learning in modeling numeric outcomes where patterns in historical data can be used to forecast future results.

One of the key indicators of success in this implementation is the seamless integration of the trained model into an interactive GUI, allowing users to input live match scenarios and receive real-time predictions. The model not only learns from the data but also respects cricket-specific rules through custom logic, such as dynamic penalty adjustments for high wicket losses and ensuring predictions are never lower than the current score.

Moreover, the project successfully handles categorical feature encoding, data normalization, and input consistency, which are critical steps in preparing real-world data for model consumption. The ability to generate meaningful outputs from both small and large datasets highlights the flexibility of the system.

Overall, the machine learning component of this project fulfills its intended purpose effectively and lays a strong foundation for future enhancements using advanced algorithms and richer datasets.

**Software Engineering Insights**

The development of the *Cricket Score Prediction using Machine Learning* project offers several valuable insights from a software engineering perspective. One of the key takeaways is the importance of modular design. The project is structured into clear functional modules: data preprocessing, model training, prediction logic, and user interface. This separation of concerns improves code readability, maintainability, and scalability, making it easier to debug and upgrade individual components without affecting the entire system.

Another insight is the role of error handling and validation in building a robust user-facing application. The use of exception handling for invalid inputs, and GUI alerts to inform users of logical errors (like selecting the same team for batting and bowling), enhances the reliability and user experience of the system.

The project also highlights the need for data quality and feature engineering in machine learning. Even the most efficient algorithms can underperform with poor or insufficient data. Therefore, software engineers must carefully prepare datasets and design meaningful features that represent real-world patterns.

Lastly, incorporating testing methodologies—such as unit and integration testing—ensures stability during updates. The project reinforces that successful machine learning applications require not only algorithmic accuracy but also sound software engineering practices for deployment and usability.

**Integration and Deployment Lessons**

The *Cricket Score Prediction using Machine Learning* project provides valuable lessons in integration and deployment, particularly in combining machine learning models with user-facing applications. One key lesson is the importance of ensuring compatibility between the model and the user interface. For example, the trained model expects a specific input format with encoded features, which must be consistently reproduced from the GUI input fields. To achieve this, robust preprocessing logic was implemented to dynamically align input data with the model’s expectations, preventing runtime errors.

Another important lesson is the necessity of error handling during integration. The system uses appropriate safeguards to manage invalid or missing user inputs, which is essential for maintaining a smooth user experience and preventing application crashes. Additionally, decoupling the core components—such as separating the machine learning logic, GUI functions, and prediction engine—greatly simplifies debugging, testing, and future upgrades.

From a deployment perspective, the project underscores the value of building a lightweight, offline-capable application using tools like Tkinter and scikit-learn. It avoids dependencies on cloud platforms or live data sources, making it easy to share and deploy across local systems. These integration and deployment practices contribute to a stable, user-friendly, and maintainable software solution.

Bottom of Form

**Limitations and Areas for Improvement**

The current *Cricket Score Prediction using Machine Learning* system, while functional and educational, has certain constraints that limit its full-scale deployment in real-world scenarios. One of the primary constraints is the limited dataset used for model training. Although 500 entries were generated to simulate realistic match conditions, the dataset does not encompass the diversity and depth of actual match situations, such as various pitch conditions, team strategies, or individual player performance.

Another key constraint lies in the basic feature set. The current model uses only a few numerical inputs—runs, wickets, overs, and runs in the last 5 overs—along with team names. It does not consider contextual cricket data such as match venue, weather conditions, batting order, or in-game momentum shifts, which are crucial for more accurate predictions.

The user interface, developed using Tkinter, is also constrained to a desktop environment and is not responsive or optimized for mobile or web usage. This limits the application’s portability and user accessibility. Moreover, the system lacks persistence—once the session ends, no user data or predictions are saved. These constraints highlight the current version as a prototype or proof of concept rather than a production-level tool for live sports analytics.

**Acknowledged Technical Limitations**

The *Cricket Score Prediction* project acknowledges several technical limitations that impact its predictive power and extensibility. A key limitation is the use of Linear Regression, which, by design, assumes a straight-line relationship between features and the target variable. Cricket outcomes, however, are influenced by nonlinear and dynamic interactions—such as sudden batting collapses, strategic bowling changes, or psychological pressure—that a linear model cannot adequately capture.

Another limitation is the lack of advanced machine learning techniques. Algorithms such as Random Forests, XGBoost, or Neural Networks would provide better predictive performance by handling more complex patterns in the data. Furthermore, the model does not currently support real-time learning or live data ingestion, making it static in nature and less effective for use during ongoing matches or tournaments.

The model also lacks feature scaling, tuning, and evaluation using standard metrics such as MAE, RMSE, or R² score, which are critical for assessing performance. Additionally, there is no model explainability layer, meaning users cannot understand which inputs most influenced a prediction—an important aspect for trust and transparency. Lastly, the absence of modular logging and exception handling makes it harder to debug or monitor the system during runtime in more complex deployments.

**Top of FormBottom of Form**

**CHAPTER – 11**

**REFERENCES**

1. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.  
   – A comprehensive guide to practical machine learning implementation in Python.
2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.  
   – Original paper introducing scikit-learn, the core library used for model training in the project.
3. McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 51–56.  
   – Documentation and explanation of the Pandas library used for data handling and preprocessing.
4. Tkinter Documentation. *TkDocs* – <https://tkdocs.com>  
   – Official reference for building GUIs using Python’s built-in Tkinter module.
5. Kumar, A., & Ravindran, B. (2008). *A machine learning approach to cricket match outcome prediction*. In *Proceedings of the International Conference on Advanced Data Mining and Applications* (pp. 552-559). Springer.  
   – Academic reference for applying ML models in cricket analytics.
6. Jain, A., & Agrawal, R. (2016). Predicting Final Score of ODI Cricket Matches: A Machine Learning Approach. *International Journal of Computer Applications*, 142(5), 15–19.  
   – Research study relevant to score prediction in limited-overs cricket using ML techniques.
7. Documentation – Python Official Site: <https://docs.python.org/3/>  
   – Reference for core Python functions and libraries used in development.