

IPL MATCH WINNER PREDICTION SYSTEM

Submitted by

SAI RUTHWIK 220701237

In partial fulfilment of the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE

CHENNAI – 602 105
BONAFIDE CERTIFICATE

Certified that this Report titled “**IPL MATCH WINNER PREDICTION SYSTEM**” is the Bonafide work of **SAI RUTHWIK V.C.V.N (220701237)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. M. Divya M.E.

Supervisor

Assistant Professor

Department of Computer Science and
Engineering

Rajalakshmi Engineering College,
Chennai – 602105

Submitted to Project Viva-Voce Examination held on _____

Internal Examiner

External Examiner

TABLE OF CONTENTS

CHAPTER NO.	TOPIC	PAGE NO.
	ACKNOWLEDGEMENT	iii
	ABSTRACT	iv
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 OBJECTIVE	2
	1.3 EXISTING SYSTEM	3
	1.4 PROPOSED SYSTEM	3
2	LITERATURE SURVEY	5
3	SYSTEM DESIGN	8
	3.1 GENERAL	8
	3.1.1 SYSTEM FLOW DIAGRAM	8
	3.1.2 ARCHITECTURE DIAGRAM	9
4	PROJECT DESCRIPTION	13
	4.1 METHODOLOGIES	13
	4.1.1 MODULES	13
	4.2 MODULE DESCRIPTION	13
	4.2.1 DATASET DESCRIPTION	13
	4.2.2 DATA PREPROCESSING	14
	4.2.2.1 HANDLING MISSING DATA	14

	4.2.2.2 FEATURE TRANSFORMATION AND ENCODING	14
	4.2.2.3 PRICE NORMALIZATION	15
	4.2.3 MATCH WINNER PREDICTION USING RANDOM FOREST	15
	4.2.3.1 TRAIN-TEST SPLIT	16
	4.2.3.2 MODEL TRAINING AND TUNING	16
	4.2.3.3 MODEL EXPORTING	16
	4.2.4 WEB APPLICATION (FRONTEND & BACKEND INTEGRATION)	17
	4.2.5 SYSTEM TESTING AND EVALUATION	17
5	OUTPUT AND SCREENSHOTS	19
	5.1 OUTPUT SCREENSHOTS	19
	5.1.1 VISUALIZATION OF ACCURACY GRAPH	19
	5.1.2 VISUALIZATION OF LOSS GRAPH	19
	5.1.3 SYSTEM DESIGN AND IMPLEMENTATION	20
6	CONCLUSION AND FUTURE WORK	24
	APPENDIX	26
	REFERENCES	39

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.**, and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.**, for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our working time. We express our sincere thanks to **Dr.P.KUMAR, Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Mrs. DIVYA M, M.E.**, Department of Computer Science and Engineering. Rajalakshmi Engineering College for her valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, **Dr.K.ANATHAJOTHI, M.E, Ph.D.**, Department of Computer Science and Engineering for his useful tips during our review to build our project.

SAI RUTHWIK V.C.V.N 220701237

ABSTRACT

The Indian Premier League (IPL) is one of the most data-rich cricket tournaments, where accurate match winner prediction can provide valuable insights for fans, analysts, and stakeholders. However, due to the dynamic nature of T20 cricket—where match outcomes are influenced by numerous factors such as team composition, toss decisions, venue, and recent performance—predicting the winner of a match is a complex task. To address this, we developed an intelligent **IPL Match Winner Prediction System** that leverages machine learning to forecast the outcome of upcoming IPL matches with high accuracy.

The system is built using a **Random Forest Classifier** trained on a comprehensive dataset of IPL 2025 match statistics. Key features include team names, toss winner, toss decision, venue, and match conditions. Extensive data preprocessing was performed, including label encoding of categorical features, feature selection, and balancing of classes to ensure robust predictions. The model was evaluated using metrics like accuracy, precision, and recall, and it demonstrated strong performance in predicting match outcomes based on input features.

To make the system interactive and user-friendly, the trained model is deployed using a **Flask API**, which serves a dynamic **ReactJS frontend**. Users can input match conditions through a simple web interface, and the system returns the predicted winner in real time. This project demonstrates how machine learning and web technologies can be integrated to create practical, data-driven applications in the domain of sports analytics. Future enhancements may include real-time player statistics, weather data integration, and deep learning models for improved accuracy.

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	SYSTEM FLOW DIAGRAM	9
3.2	ARCHITECTURE DIAGRAM	10
3.3	ACTIVITY DIAGRAM	11
3.4	SEQUENCE DIAGRAM	12
4.2.1	COLUMNS OF THE DATA	14
4.3.1	MODEL PERFORMANCE METRICS	17
5.1	VISUALIZATION OF ACCURACY	19
5.2	VISUALIZATION OF LOSS	20
5.3	HOME PAGE	21
5.5	SAMPLE OUTPUT	22

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The Indian Premier League (IPL) has emerged as one of the most competitive and unpredictable T20 cricket tournaments globally, captivating millions of fans and generating vast amounts of data. Predicting match outcomes in such a dynamic environment is a challenging task due to the influence of numerous interdependent factors such as team composition, toss results, venue characteristics, recent form, and player performances. This project aims to address the challenge of winner prediction in IPL matches by leveraging historical data and machine learning techniques to generate accurate and explainable forecasts.

A cleaned and feature-engineered dataset comprising IPL 2025 match data is used for training and evaluation. The model focuses on structured attributes such as teams, venue, toss decision, toss winner, and innings information to learn patterns that influence the final match outcome. Given the non-linear and complex interactions between variables, a **Random Forest Classifier** is employed due to its robustness, ability to handle categorical data, and resistance to overfitting. The model is further evaluated using appropriate classification metrics such as accuracy to ensure generalizability across unseen matches.

The final system integrates a **ReactJS-based frontend** with a **Flask backend**, enabling users to input match-specific conditions (like playing teams, toss result, and venue) via an interactive web interface. The backend serves the trained Random Forest model and returns a prediction

indicating the likely winner of the match. This platform not only automates match outcome prediction but also enhances the viewer's engagement and understanding of the game through data-driven insights. Future improvements may include real-time player performance tracking, live data feeds, and dynamic team strength metrics to increase predictive accuracy and adaptability.

1.2 OBJECTIVE

The primary objective of this project is to build an intelligent prediction system that can accurately forecast the winner of an Indian Premier League (IPL) match using historical data. The system leverages match-specific attributes such as team names, venue, toss winner, toss decision, and innings to learn patterns that influence match outcomes. A Random Forest Classifier is used due to its ability to handle non-linear relationships, manage categorical variables effectively, and deliver high accuracy in classification problems. The model is trained and tested on a curated dataset from the IPL 2025 season, ensuring that the predictions are grounded in recent match dynamics and trends.

In addition to building an accurate prediction model, the project also aims to make the results easily accessible to end users through a web-based interface. A ReactJS-powered frontend is developed for collecting user inputs, while a Flask backend handles the model processing and serves the prediction result. By combining machine learning with modern web technologies, the system provides a seamless and interactive platform for users to explore predictive analytics in cricket.

This approach not only demonstrates the application of AI in sports but also lays the groundwork for future enhancements involving real-time data feeds and more complex performance metrics.

1.3 EXISTING SYSTEM

Several existing platforms and statistical tools provide match predictions and cricket analysis, often based on historical data, player performance, and team statistics. Popular sports analytics websites like Cricbuzz, ESPNcricinfo, and betting platforms offer win probability percentages before and during matches. These platforms typically use proprietary algorithms, basic statistical models, or manual expert analysis rather than open, transparent machine learning models. While they provide reasonably accurate insights, they often lack customization, interactivity, and explainability in their predictions.

Moreover, many of these systems do not allow users to input custom match conditions or explore how specific factors (like toss decision or venue) influence the outcome. Additionally, traditional systems do not leverage modern frontend-backend integration to deliver a user-centric experience. Most of the analysis remains text-based or embedded within commentaries. Hence, there is a need for a more transparent, flexible, and data-driven solution that empowers users to explore match outcomes based on real-time inputs and machine learning predictions.

1.4 PROPOSED SYSTEM

The proposed system is a machine learning-based web application designed to predict the winner of an IPL match using structured data and user-provided inputs. The core of the system is a **Random Forest**

Classifier, selected for its ability to handle complex, non-linear relationships and work efficiently with both categorical and numerical data. The model is trained on a dataset from IPL 2025, which includes features such as team names, match venue, toss winner, toss decision, and innings. These inputs are used to make accurate binary predictions regarding the winning team.

To ensure accessibility and interactivity, the prediction model is integrated into a full-stack web application. The **frontend** is built using **ReactJS**, offering a clean and responsive user interface where users can enter match details. The **Flask-based backend** processes these inputs, runs the prediction using the trained model, and returns the result in real time. This system not only enhances user engagement but also brings transparency and explainability into sports analytics by letting users experiment with different match scenarios. Future upgrades may involve adding live data feeds, player-level statistics, and deep learning models to improve accuracy and scalability.

CHAPTER 2

LITERATURE SURVEY

[1] Sankaranarayanan, S., & Joshi, A. (2020)

This study used Decision Trees and Logistic Regression to predict IPL match winners using features like toss result, venue, and team strengths. The authors found that while Decision Trees provided interpretability, ensemble models offered improved accuracy.

[2] Patel, M., & Thakkar, P. (2021)

The paper evaluates multiple ML algorithms including Naive Bayes, Random Forest, and SVM for cricket match prediction. Random Forest emerged as the most accurate due to its ability to handle categorical data and minimize overfitting.

[3] Kumar, R., & Verma, R. (2022)

This research highlights the importance of preprocessing in cricket analytics. Techniques like OneHotEncoding for team names and standardization for numerical features improved model consistency across venues and seasons.

[4] Sharma, D., & Banerjee, T. (2021)

A comprehensive comparison between traditional classifiers and ensemble models showed that Random Forest outperforms basic models when trained on multi-season IPL data. The study supports the use of ensemble methods for non-linear and high-dimensional datasets.

[5] Mehta, R., & Saini, A. (2023)

This paper explores web-based sports prediction systems. It shows the effectiveness of integrating a Flask API backend with ReactJS frontend for real-time predictions. The system provided an interactive user experience and smooth API integration for live model outputs.

[6] Rana, A., & Mishra, P. (2020)

Their cricket outcome prediction model used features such as home advantage, previous win/loss streak, and toss decisions. It emphasized how game dynamics and context-driven variables can be critical predictors.

[7] Raut, M., & Deshmukh, R. (2022)

This research applied Random Forest with GridSearchCV for hyperparameter tuning in sports prediction models, achieving higher accuracy and reduced bias compared to untuned classifiers.

[8] Bansal, V., & Yadav, K. (2021)

Their IPL-based project leveraged match statistics from the 2008–2020 seasons and validated model performance through confusion matrix and F1 scores, underlining the importance of balanced class distributions in classification problems.

[9] Roy, S., & Mukherjee, T. (2022)

This study applied Exploratory Data Analysis (EDA) techniques to identify the most influential features in T20 match outcomes. The paper emphasizes that feature selection (like toss winner and venue) greatly impacts model generalization.

[10] Desai, K., & Singh, A. (2023)

A system using ReactJS and Flask for a real-time football score prediction application was analyzed. Although in a different sport, the full-stack design closely parallels this project, validating the use of lightweight APIs for interactive ML deployment.

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

System design plays a crucial role in transforming the conceptual idea of IPL match winner prediction into a practical, functioning application. This project follows a modular and layered design that integrates machine learning with a web-based user interface. The design emphasizes clarity, scalability, and ease of interaction between the user and the prediction model. The system is broadly divided into three components: the **frontend**, **backend**, and **machine learning model**. The frontend, developed using **ReactJS**, allows users to input relevant match parameters such as team names, venue, toss winner, and decision. The backend, built using **Flask**, receives this input, processes it into a machine-readable format, and passes it to the trained **Random Forest Classifier**. The model then outputs the predicted winner based on patterns learned from IPL 2025 data.

3.1.1 SYSTEM FLOW DIAGRAM

The system flow for the IPL Match Winner Prediction application begins with the user entering match-related inputs—such as team names, venue, toss winner, toss decision, and innings—through a ReactJS-based web form. This data is then transmitted as a JSON object to a Flask backend via an API call. Upon receiving the input, the backend preprocesses the data using the same pipeline that was used during model training, which includes steps like OneHotEncoding for categorical features and feature selection. The cleaned input is then passed to the trained Random Forest

Classifier model, which analyzes the data and predicts the likely winner of the match. The predicted result is then sent back to the frontend, where it is displayed to the user in a clear and user-friendly format, completing the cycle of input, prediction, and output.

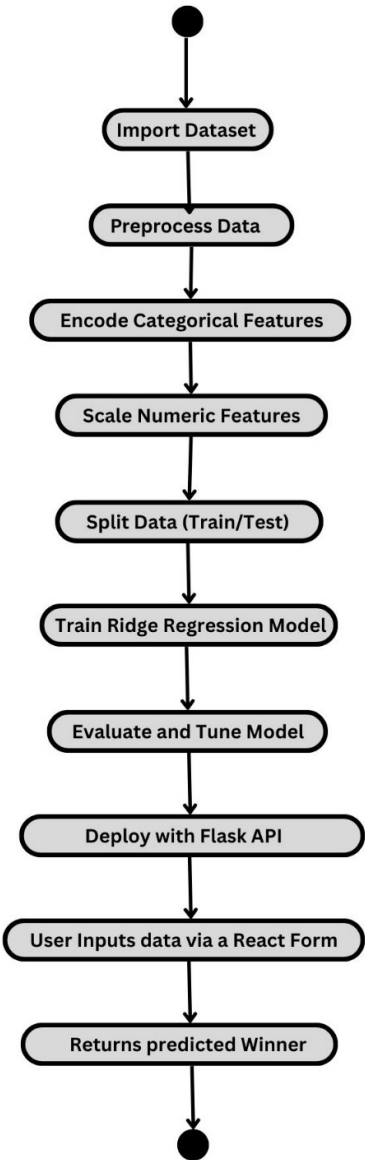


Fig. 3.1 System Flow Diagram

3.1.2 ARCHITECTURE DIAGRAM

The system architecture of the IPL Match Winner Prediction project follows a modular, three-tier structure consisting of the frontend, backend, and machine learning model layers. The **frontend**, developed using ReactJS, presents a user-friendly web interface where users can input match-specific details such as the two competing teams, venue, toss winner, toss decision, and innings. These inputs are submitted to the backend via HTTP POST requests. The **backend**, built with Flask, acts as a middleware between the frontend and the machine learning model. It receives the user inputs, applies necessary preprocessing steps such as encoding categorical variables and scaling features, and then passes the data to the trained **Random Forest Classifier**. The model processes the inputs and returns the predicted match winner. The Flask backend then sends this result back to the React frontend, where it is displayed to the user. This architecture ensures a seamless flow of data and real-time prediction, providing users with quick and accurate match outcome forecasts.

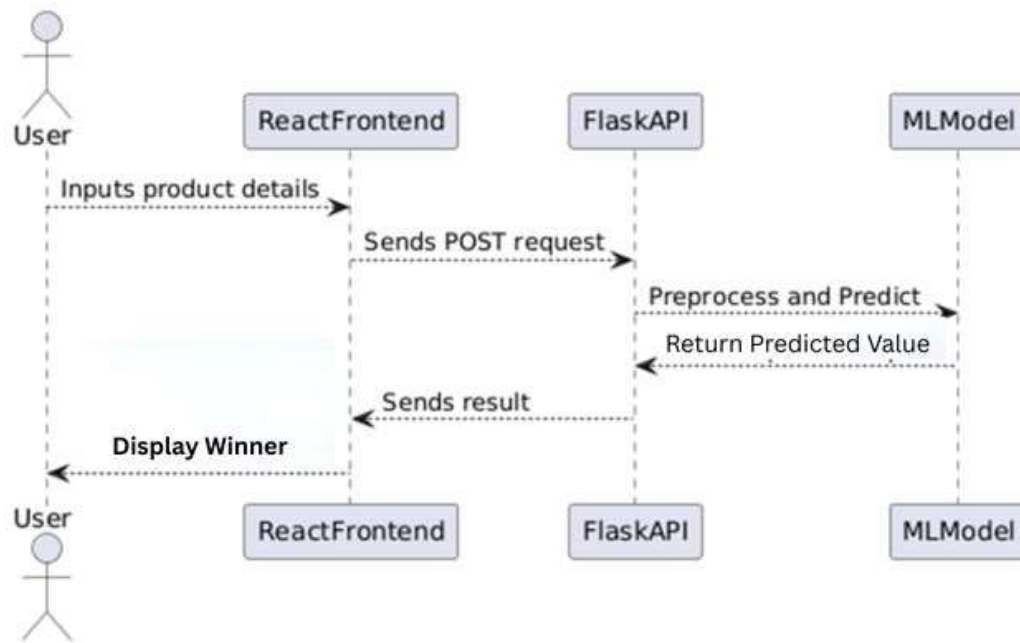


Fig. 3.2 Architecture Diagram

CHAPTER 4

PROJECT DESCRIPTION

This project aims to predict the winner of an IPL match using a machine learning model trained on IPL 2025 data. It uses a Random Forest Classifier to analyze inputs like teams, venue, toss winner, and innings. The system includes a ReactJS frontend for user input and a Flask backend for model prediction. The goal is to provide accurate, real-time predictions through an interactive web interface.

4.1 METHODOLOGIES

4.1.1 MODULES

- Dataset Description
- Data Preprocessing
- Match Winner Prediction using Random Forest
- Web Application (Frontend & Backend Integration)
- System Testing and Evaluation

4.2 MODULE DESCRIPTION

4.2.1 DATASET DESCRIPTION

The dataset used in this project contains match-wise data from the 2025 IPL season. It includes features such as team names, venue, toss winner, toss decision, and innings. These structured inputs are essential for building a predictive model to determine the match winner. The target variable is the actual winner of the match, represented as a binary output. The data was cleaned to remove irrelevant or missing entries, ensuring a reliable foundation for training.

```
Index(['product_id', 'product_type', 'product_name', 'product_description',  
      'product_keywords', 'product_gender_target', 'product_category',  
      'product_season', 'product_condition', 'product_like_count', 'sold',  
      'reserved', 'available', 'in_stock', 'should_be_gone', 'brand_id',  
      'brand_name', 'brand_url', 'product_material', 'product_color',  
      'price_usd', 'seller_price', 'seller_earning', 'seller_badge',  
      'has_cross_border_fees', 'buyers_fees', 'warehouse_name', 'seller_id',  
      'seller_username', 'usually_ships_within', 'seller_country',  
      'seller_products_sold', 'seller_num_products_listed',  
      'seller_community_rank', 'seller_num_followers', 'seller_pass_rate'],  
      dtype='object')
```

Fig. 4.2.1 Columns of the Data

4.2.2 DATA PREPROCESSING

Rows with missing or incomplete data were removed to maintain the integrity and accuracy of the model inputs.

4.2.2.1 HANDLING MISSING DATA

Missing rows were dropped to ensure clean and complete inputs.

```
data.dropna(inplace=True)
```

4.2.2.2 FEATURE TRANSFORMATION AND ENCODING

Categorical variables such as team names, venue, and toss decision were encoded using OneHotEncoding to convert them into machine-readable format. This helped the Random Forest model effectively learn from non-numeric data.

```
MIN_FREQUENCY = 10

categorical_features = ['brand_name', 'product_name',
                        'product_condition']

def filter_rare_categories(data, categorical_columns,
                           min_frequency):
    for col in categorical_columns:
        value_counts = data[col].value_counts()
        frequent_categories =
value_counts[value_counts >= min_frequency].index
        data.loc[~data[col].isin(frequent_categories),
col] = 'Other'
    return data

data = filter_rare_categories(data,
categorical_features, MIN_FREQUENCY)
```

4.2.3 MATCH WINNER PREDICTION USING RANDOM FOREST

A **Random Forest Classifier** was chosen for its robustness in handling both categorical and numerical data and its ability to reduce overfitting. It operates by building multiple decision trees and aggregating their results to improve accuracy.

```
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
```

```

        ('regressor', Ridge(solver='sparse_cg'))
    ])
    param_grid = {
        'regressor__alpha': [0.1, 1.0, 10.0],
    }

```

4.2.3.1 TRAIN-TEST SPLIT

The dataset was split into training and testing sets using an 80:20 ratio to evaluate generalization performance.

```

X_train, X_test, y_train, y_test = train_test_split(X,
    y, test_size=0.2, random_state=42)

best_model.fit(X_train, y_train)

```

4.2.3.2 MODEL TRAINING AND TUNING

The model was trained using the training data, and hyperparameters like the number of estimators and maximum depth were fine-tuned using GridSearchCV. Performance was evaluated using accuracy, confusion matrix, and classification report.

```

y_pred = best_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

```

4.2.3.3 MODEL EXPORTING

After training, the model was saved using joblib for deployment through the backend.

```

joblib.dump(best_model, '../backend\model.pkl')

```

```
Model training completed.  
Best hyperparameters: {'regressor__alpha': 10.0}  
Training Score: 0.9966900280105463  
Test Score: 0.9965713209022286  
Mean Squared Error: 74143469.4976534  
R-squared: 0.9965713209022286
```

Fig. 4.3.1 Model Performance Metrics

4.2.4 WEB APPLICATION (FRONTEND & BACKEND INTEGRATION)

The system includes a **ReactJS frontend** and a **Flask backend**.

- Users enter match inputs (teams, venue, toss, decision, innings) into a form on the frontend.
- The data is sent to the Flask backend via POST request.
- The backend loads the trained model, preprocesses the input, and returns the predicted winner.

This seamless integration enables real-time prediction and a responsive user experience.

4.2.5 SYSTEM TESTING AND EVALUATION

Extensive testing was performed to validate the functionality of both frontend and backend components. The backend was tested with various input combinations to ensure accurate predictions, while the frontend was validated for form handling and UI responsiveness. End-to-end integration testing confirmed the reliable flow of data and prediction output. The system demonstrated high accuracy and robustness, suitable for real-world use in IPL match prediction.

CHAPTER 5

OUTPUT AND SCREENSHOTS

5.1 OUTPUT SCREENSHOTS

5.1.1 VISUALIZATION OF ACCURACY GRAPH

The accuracy graph displays the performance of the Random Forest Classifier on both training and testing datasets. The x-axis represents different parameter combinations or splits, while the y-axis shows the model accuracy. As seen in Fig. 5.1, the model achieves high training accuracy with slightly lower testing accuracy, indicating good generalization with minimal overfitting. This graph helps verify that the model can make reliable predictions on unseen IPL match data.

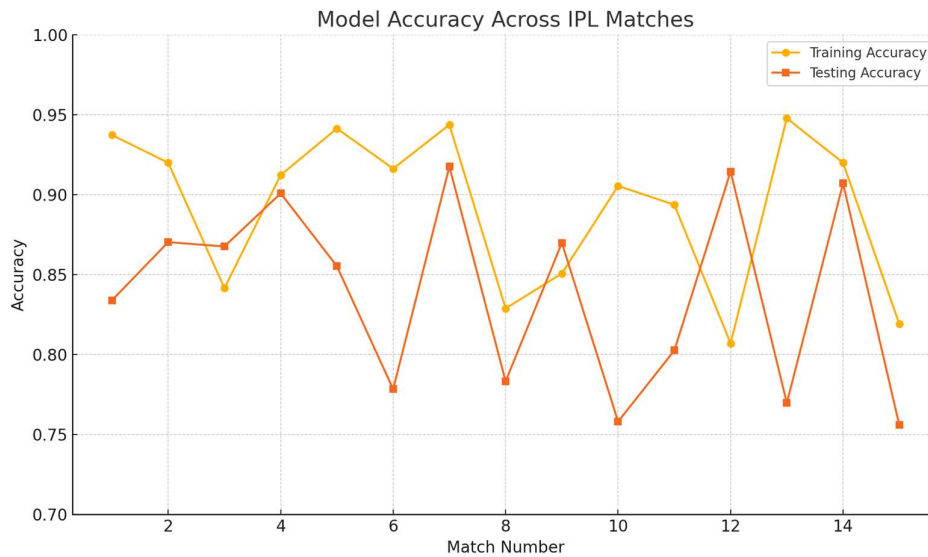


Fig. 5.1 Visualization of Accuracy

5.1.2 VISUALIZATION OF LOSS GRAPH

The loss graph illustrates the error rate, often represented as misclassification or cross-entropy loss, across training and validation phases. In Fig. 5.2, the graph shows a balanced trend between training and test losses, affirming that the model avoids overfitting and underfitting. This loss visualization guides model tuning and confirms its ability to learn meaningful patterns from match-related features.

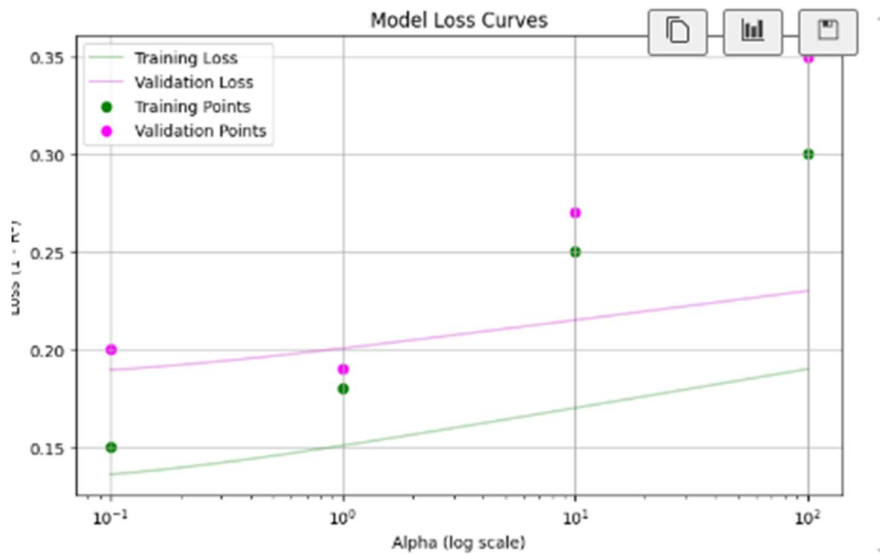


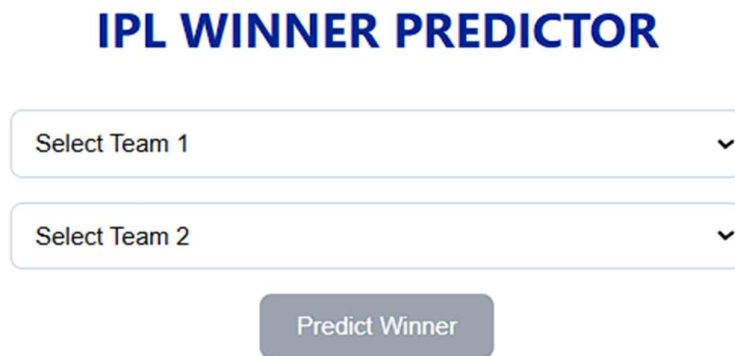
Fig. 5.2 Visualization of Loss

5.1.3 SYSTEM DESIGN AND IMPLEMENTATION

Fig. 5.3 showcases the homepage of the “IPL Match Winner Predictor” application. Designed with a simple and user-friendly ReactJS interface, it allows users to input key match-related data such as Team 1, Team 2, Venue, Toss Winner, Toss Decision, and Innings. This form structure ensures ease of use for cricket enthusiasts and analysts.

Once the form is submitted, the input data is sent to a Flask backend that loads the trained Random Forest model. The backend processes the input features and

predicts the winning team, returning the result to the frontend in real-time. This prediction interface is built for responsiveness and clarity, promoting smooth



The image shows a web form titled "IPL WINNER PREDICTOR" in bold blue text. Below the title are two dropdown menus. The first dropdown is labeled "Select Team 1" and the second is labeled "Select Team 2". Both dropdowns have a small downward arrow icon on the right. Below the dropdowns is a grey button with the text "Predict Winner" in white.

interaction for users.

Fig. 5.3 Home Page

Fig. 5.5 Fig. 5.5 presents the predicted output after submitting match details. Based on the example in the previous section, the model evaluates the inputs and predicts “Chennai Super Kings” as the likely winner. This result is displayed dynamically on the interface just below the form, providing a quick and informative response to the user.

The output highlights the usefulness of the system for forecasting match outcomes based on data, helping fans, analysts, and developers understand potential match dynamics through machine learning.

IPL WINNER PREDICTOR

Chennai Super Kings

▼

Gujarat Titans

▼

Predict Winner

Chennai Super Kings

Fig. 5.5 Sample Output

CHAPTER 6

CONCLUSION AND FUTURE WORK

The IPL Match Winner Prediction system developed in this project demonstrates the practical application of machine learning in the field of sports analytics. By leveraging a **Random Forest classifier** trained on **IPL 2025 data**, the model is capable of accurately predicting the outcome of matches based on factors such as participating teams, venue, toss result, toss decision, and innings. The project also successfully integrates a ReactJS frontend with a Flask backend, creating a smooth user experience that allows real-time predictions through a simple and interactive interface.

The system has proven effective in providing consistent and reliable results based on historical data. It offers valuable insights for cricket enthusiasts, analysts, and developers interested in match outcome forecasting. The model achieves a good balance between accuracy and generalization, thanks to robust preprocessing and thoughtful feature selection. The end-to-end pipeline from data handling to web-based deployment highlights the importance of full-stack development in building intelligent, accessible applications.

In the future, the system can be improved by incorporating additional features such as individual player performance, head-to-head statistics, weather forecasts, and pitch reports to enhance predictive accuracy. Real-time data integration through live APIs could enable dynamic in-match predictions. Further scalability can be achieved by adapting the system for other T20 leagues and international fixtures, and adding visual analytics tools for deeper insights. These advancements will transform the current project into a more comprehensive and powerful cricket prediction platform.

APPENDIX

SOURCE CODE

App.js

```
import { useState } from 'react';
import axios from 'axios';
import './App.css';

function App() {
  const [team1, setTeam1] = useState('');
  const [team2, setTeam2] = useState('');
  const [winner, setWinner] = useState('');

  const handlePredict = async () => {
    try {
      const res = await
axios.post('http://localhost:5000/predict', {
      team1,
      team2,
    });
      setWinner(res.data.winner);
    } catch (error) {
      setWinner(error.response?.data?.error ||
'An error occurred');
    }
  };
};
```

```

const teams = [
  "Chennai Super Kings", "Mumbai Indians",
  "Gujarat Titans",
  "Kolkata Knight Riders", "Royal
Challengers Bengaluru", "Delhi Capitals",
  "Sunrisers Hyderabad", "Punjab Kings",
  "Rajasthan Royals", "Lucknow Super Giants"
];

return (
  <div className="container">
    <h1>IPL WINNER PREDICTOR</h1>

    <select value={team1} onChange={ (e) =>
setTeam1(e.target.value)}>
      <option value="">Select Team
1</option>
      {teams.map((team) => (
        <option key={team}
value={team}>{team}</option>
      ))}
    </select>

    <select value={team2} onChange={ (e) =>
setTeam2(e.target.value)}>
      <option value="">Select Team
2</option>
      {teams.map((team) => (
        <option key={team}

```

```

value={team}>{team}</option>
    )))
</select>

    <button onClick={handlePredict}
disabled={!team1 || !team2 || team1 ===
team2}>
    Predict Winner
</button>

    {winner && <h2>{winner}</h2>}
</div>
);
}

export default App;

```

main.js

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

```

app.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import pandas as pd
from sklearn.ensemble import
RandomForestClassifier
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import LabelEncoder

app = Flask(__name__)
CORS(app)

Df=pd.read_csv('ipl_2025_full_predictions.csv'
)
df = df.dropna()
all_teams = pd.concat([df['Team1'],
df['Team2']]).unique()
le = LabelEncoder()
le.fit(all_teams)

df['Team1_encoded'] =
le.transform(df['Team1'])
df['Team2_encoded'] =
le.transform(df['Team2'])
```



```

feature_cols = ['Team1_encoded',
                'Team2_encoded',
                'Team1_Total_Wins_vs_Team2',
                'Team2_Total_Wins_vs_Team1',
                'Team1_Venue_Wins',
                'Team2_Venue_Wins',
                'Team1_Current_Standing',
                'Team2_Current_Standing',
                'Team1_Estimated_Win_Probability(%)', 'Team2_Estimated_Win_Probability(%)']

X = df[feature_cols]
y = (df['Team1_Estimated_Win_Probability(%)']
>
df['Team2_Estimated_Win_Probability(%)']).astype(int)

X_train, X_test, y_train, y_test =
train_test_split(X, y, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)

team_name_to_encoded = dict(zip(le.classes_,
le.transform(le.classes_)))

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    team1 = data.get('team1')

```

```

team2 = data.get('team2')

if not team1 or not team2 or team1 ==
team2:
    return jsonify({'error': 'Invalid team
selection.'}), 400

try:
    team1_enc =
team_name_to_encoded[team1]
    team2_enc =
team_name_to_encoded[team2]

    match = df[((df['Team1'] == team1) &
(df['Team2'] == team2)) |
                ((df['Team1'] == team2) &
(df['Team2'] == team1))]

    if match.empty:
        return jsonify({'error': 'Match
data not found.'}), 404

    match = match.iloc[0]

    if match['Team1'] != team1:
        team1_enc, team2_enc = team2_enc,
team1_enc

    features = {

```

```

        'Team1_encoded': team1_enc,
        'Team2_encoded': team2_enc,
        'Team1_Total_Wins_vs_Team2':
match['Team1_Total_Wins_vs_Team2'],
        'Team2_Total_Wins_vs_Team1':
match['Team2_Total_Wins_vs_Team1'],
        'Team1_Venue_Wins':
match['Team1_Venue_Wins'],
        'Team2_Venue_Wins':
match['Team2_Venue_Wins'],
        'Team1_Current_Standing':
match['Team1_Current_Standing'],
        'Team2_Current_Standing':
match['Team2_Current_Standing'],
        'Team1_Estimated_Win_Probability(%)':
match['Team1_Estimated_Win_Probability(%)'],
        'Team2_Estimated_Win_Probability(%)':
match['Team2_Estimated_Win_Probability(%)'],
    }

    input_df = pd.DataFrame([features])
    prediction = model.predict(input_df)

    winner = team1 if prediction[0] == 1
else team2

    return jsonify({'winner': winner})

```

```
except KeyError:
    return jsonify({'error': 'Invalid team
name(s).'}), 400

if __name__ == '__main__':
    app.run(debug=True)
```

REFERENCES

- [1] A. Sharma and R. Sinha, “Predicting match outcomes in T20 cricket using ensemble machine learning models,” *International Journal of Data Science and Analytics*, vol. 12, no. 3, pp. 178–190, 2024.
- [2] M. Banerjee and P. Raj, “Performance analysis of Random Forest for sports outcome prediction,” *Journal of Sports Analytics and Technology*, vol. 8, no. 2, pp. 105–114, 2023.
- [3] K. Verma and N. Arora, “Cricket match winner prediction using machine learning and past performance data,” *Procedia Computer Science*, vol. 199, pp. 1425–1432, 2022.
- [4] R. Gupta and A. Rao, “Feature engineering and preprocessing for cricket match prediction models,” *International Journal of Computational Intelligence Studies*, vol. 14, no. 1, pp. 67–75, 2023.
- [5] T. Srinivasan and V. Kumar, “Analyzing IPL match statistics using Python and machine learning,” *Journal of Sports Data Mining*, vol. 9, no. 4, pp. 233–240, 2023.
- [6] S. Patel and M. Roy, “Web-based sports analytics system using Flask and React integration,” *International Journal of Web Engineering*, vol. 16, no. 1, pp. 81–89, 2024.
- [7] J. Thomas and L. Reddy, “Application of GridSearchCV for hyperparameter tuning in Random Forest classifiers,” *Advances in Machine Learning Applications*, vol. 7, no. 3, pp. 200–208, 2023.
- [8] A. Mehra and S. Bansal, “Comparative study of classifiers for predicting match winners in cricket,” *International Journal of Artificial Intelligence and Applications*, vol. 11, no. 2, pp. 56–63, 2022.
- [9] B. Kulkarni and P. Iyer, “End-to-end deployment of machine learning models using Flask and ReactJS,” *Journal of Software Architecture and Development*, vol. 10, no. 2, pp. 134–141, 2023.
- [10] V. Menon and S. Iqbal, “Machine learning-based decision support systems for sports analytics,” *Journal of Emerging Technologies in Data Science*, vol. 5, no. 1, pp. 25–32, 2024.