Infosys Internship 5.0

**Project Documentation:**

**Title: IPL\_INFOGRAPHICS**

**Introduction**

Predicting IPL match outcomes is challenging due to the game's dynamic nature and various influencing factors such as player form, team composition, and match conditions.

The IPL Infographics Visualization project analyzes IPL data from 2008 to 2023 to identify scoring trends and key performance metrics. It uses regression models to predict winning and losing probabilities and classification techniques to uncover team performance patterns. Forecasting methods are applied to predict future results based on historical data.

The project starts with data preprocessing to clean and organize the raw dataset, followed by extensive Exploratory Data Analysis (EDA) to uncover trends and insights. Advanced machinelearning models are then applied, with a focus on hyperparameter tuning for optimal performance.

The project presents complex IPL data through clear, visually engaging infographics, empowering cricket analysts, team strategists, and enthusiasts to gain valuable insights. An interactive UI dashboard built with Streamlit allows users to explore visualizations, predictions, and key insights, making the analytics accessible and engaging for a wide audience.

**Problem Statement:**

Cricket enthusiasts, analysts, and teams often struggle to predict match outcomes, evaluate team performance trends, and forecast future match results accurately due to the complexity and volume of IPL data. Existing tools lack an intuitive interface that integrates predictive analytics with actionable insights, leaving users unable to make data-driven decisions effectively.

The IPL Predictions app aims to address this challenge by leveraging machine learning models to provide winning and losing probabilities, cumulative run calculations, and future match point forecasts based on historical IPL data, all presented through an interactive and user-friendly interface.

**Abstract**

Predicting the outcome of an IPL match is a complex task due to the dynamic and multifaceted nature of the game, where factors such as player form, team composition, match conditions, and scoring patterns significantly influence the results. The IPL Infographics Visualization project addresses this challenge by analyzing IPL data from 2008 to 2023, focusing on identifying scoring trends and predicting match outcomes.

Using regression models, the project calculates cumulative runs at various stages of the game and forecasts the probabilities of winning or losing. These insights are presented through visually appealing and intuitive infographics, offering a clear understanding of match dynamics and team performance. This project empowers analysts, strategists, and cricket enthusiasts by providing actionable insights and enhancing data-driven decision-making in the context of IPL matches.

**Project Mentor:** Mr. Amal Salilan

Provided expert guidance, offered valuable insights, and helped the team stay focused on project goals.

**Team Members:**

* Reshma S
* Gopikrishna V
* Sagar Thube
* Sampath Kumar Illa
* Sai Hitesh Kannepelly
* Aditya Raj
* Lekkala Mahima Sai
* Govinda Prasad D
* Nagira R
* Dwarampudi Aasritha Reddy
* Sai Rahul N
* Navya Sree Gouru
* Nikita Avhad
* Kayala Pavan kumar
* SRIMANNARAYANA ADAPA
* KANDULA AKSHYA
* Allada Nomiya
* Bhavithra R S

**Project Scope**

The IPL Infographics Visualization project aims to analyze IPL match data from 2008 to 2023 to provide predictive insights and visualization tools. This scope encompasses:

**Inclusions:**

* Development of an interactive user interface for IPL score predictions and match forecasting.
* Features for data filtering and exporting to generate customized predictions.
* Timeframe of analysis covering matches from 2008 to 2023.

**Exclusions:**

* The project does not provide live scoring or real-time match updates; it focuses exclusively on historical data and predictive analysis.
* Complex external data sources or deep feature engineering beyond basic player statistics and match outcomes.
* Integration with external financial databases for real-time economic updates or financial metrics.

**Constraints:**

* Project is limited to using publicly available datasets.
* The system requires an active internet connection for real-time interactions with the interface and AI models.

**Requirements**

**Functional Requirements:**

* Display IPL Match Data and Trends Visually.
* Provide Data-Driven Responses to User Queries via user-interface.
* Enable Users to Filter Data by Team, Player, Season, and Match Type.

**Non-Functional Requirements:**

* User-Friendly Interface with Responsive Design.
* Reliable performance under high user load.
* Secure handling of user queries.

**User Stories:**

1. As a cricket analyst, I want to compare IPL team performance across seasons to identify trends and make data-driven recommendations for team strategies.
2. As a team strategist, I want to compare the performance of different players in various match conditions (e.g., home vs. away, day vs. night) to optimize team selection.
3. As a cricket fan, I want to query specific IPL player statistics (e.g., runs, wickets, strike rate) to follow my favorite players and their performances.
4. As a casual viewer, I want to understand how my team's performance compares to others, especially in terms of match outcomes and player contributions.
5. As a sports journalist, I want to generate reports on IPL match statistics (e.g., run rates, partnerships) to include in articles and media coverage.

**Technical Stack**

**Programming Languages:** Python

**Frameworks/Libraries:** Streamlit, Pandas, NumPy, Matplotlib, Seaborn ,Scikit -learn (sklearn), plotly  
**Databases:** CSV-based datasets  
**Tools/Platforms:** Visual Studio code, Jupyter notebook, Google colab,GitHub

**Other:** Pickle (for model saving)

**Architecture/Design**

**System Architecture Overview:**

1. **Data Layer:**
   * **Stores and manages IPL match data** Historical data about IPL matches, including player statistics, match results, and team performance, is stored in CSV files or a database (if scaling).
   * **Data Management:** Data is processed using Python libraries like pandas to load, clean, and inspect the dataset.
2. **Processing Layer:**
   * **Data Cleaning and Preprocessing:**Python (pandas, scikit-learn) is used for data cleaning, handling missing values, transforming columns, and feature engineering (e.g., calculating player averages, team statistics).
   * **Feature Engineering**: Relevant features such as batting and bowling averages, player performance across matches, and team form are created to improve prediction accuracy.
   * **Modeling**: Models such as RandomForest, XGBoost, or Logistic Regression are trained using the processed data to predict match outcomes or player performance. Hyperparameter tuning (using GridSearchCV or RandomizedSearchCV) is performed to optimize model performance.
   * **Model Evaluation:** Scikit-learn is used to evaluatemodels with metrics like accuracy, precision, recall, F1-score for classification, or RMSE for regression.The best model is selected based on performance and is saved for deployment.
3. **Presentation Layer:**
   * **Streamlit for Interactive Web App Interface:** Streamlit is used to build a user-friendly interface where users can input IPL match data (teams, players, conditions) and get real-time predictions on match outcomes or player performance

**Key Design Decisions:**

* Streamlit was selected for its rapid prototyping capabilities and ease of deployment, allowing an interactive web app for IPL predictions.

**Trade-offs:**

* CSV files are used initially for simplicity in handling the data.

**Development**

**Technologies Used:**

**Streamlit**: for the front-end interface.

**Plotly**: For creating interactive visualizations.

**Scikit-learn**: For machine learning and predictive modeling.

**Pandas & NumPy:** For data manipulation and analysis.

**VS Code & Jupyter Notebook:** For code development and analysis.

**Coding Standards:**

**VScode**: for Python scripts.

**Modular Code Structure**: Ensuring scalability and readability.

**Challenges:**

**Integration:** Ensuring smooth integration between Streamlit and Plotly.

**Model Optimization**: Fine-tuning machine learning models for accurate predictions.

**Data Consistency:** Handling data formatting inconsistencies and missing values.

**Solutions:**

**Standardized Preprocessing Pipeline**: For consistent data inputs.

**Iterative Testing and Feedback Loops**: For continuous model and interface optimization.

**Data Validation and Cleaning**: To address and fix inconsistencies in the dataset.

**Testing**

**Testing Approach:**

* **Unit Tests:**

Test individual components like data preprocessing, ML models, and Streamlit UI.

* **Integration Tests:**

Ensure smooth interaction between ML pipeline, UI, and APIs.

* **System Tests:**

Validate app functionality in real-world scenarios with multiple users.

**2. Test Results**

* **Accuracy**: Achieved 97% accuracy for match predictions.
* **Usability**: UI elements load within 2 seconds; no significant bugs found.
* **Performance**: Handled 100 concurrent users with predictions under 1 second.
* **Reliability**: Handled errors gracefully

**Deployment**

**Process:**

1. **Host the Streamlit App on Streamlit Cloud**: Deploy the Streamlit app on Streamlit Cloud to make the interactive interface publicly accessible for IPL match predictions.
2. **Integrate Machine Learning Pipeline into Streamlit:** Incorporate the machine learning model (e.g., Random Forest, XGBoost) that predicts match-winning probabilities and forecasts cumulative runs based on historical IPL data into the Streamlit app. The model predicts match outcomes, team performance, and player stats by processing user inputs (e.g., teams, players, match conditions).
3. **Forecasting using Historical Data:** The app should use the machine learning pipeline to forecast match outcomes and predict cumulative runs for upcoming matches based on past performance and data trends.
4. **Ensure Model Efficiency and Deployment:** Make sure the model is efficiently trained and saved (using Pickle or Joblib) to be loaded in the Streamlit app for real-time predictions.

**Automation:**

* **Deployment Scripts:**Create deployment scripts ( Streamlit Cloud configuration) to automate the process of app updates, ensuring smooth transitions between local and cloud environments.Set up automated deployment for model updates to reflect improvements or changes in the pipeline.

**Instructions for Deployment:**

1. **Clone the Repository:** Clone the project repository to your local machine or remote environment.
2. **Set up Environment:** Create and activate a virtual environment for the project (if needed)
3. **Install Dependencies:** Install the required Python libraries using pip (make sure to include libraries for ML models and Streamlit).
4. **Prepare Model Files:** Ensure the trained model (e.g., Pickle or Joblib file) is placed in the appropriate directory.
5. **Run Streamlit App Locally**: Launch the app on your local machine for testing before deploying to Streamlit Cloud.
6. **Deploy on Streamlit Cloud**: After testing locally, deploy the app to Streamlit Cloud by pushing the code to a repository (e.g., GitHub) and connecting the repository to Streamlit Cloud.
7. Automate Updates (Optional): Set up a CI/CD pipeline to automate future updates of the app when new data is available or models are improved.

**User Guide**

**Setup Instructions:**

1. Navigate to the app URL provided.
2. Use the interactive user interface for analyze the data

* For to find the cumulative runs go to the score prediction option in menu bar
* For to find the probability of winning and losing go to the winner prediction option in menu bar
* For to find the future match points of a particular team go to the forecast option in menu bar

**Troubleshooting Tips:**

* Ensure a stable internet connection
* Contact support for data-related issues.

**Conclusion**

The **IPL Predictions** project successfully provides:

* **Interactive Dashboards:** The Interactive User Interface (UI) of the IPL Predictions app enables users to input team and player data to predict match outcomes, displaying winning/losing probabilities. It calculates cumulative runs and visualizes team performance trends across seasons. The UI also forecasts future match points and presents insights through interactive charts and graphs. The design is intuitive, providing a seamless experience for both fans and analysts.
* **AI-powered Insights:** Leveraging machine learning models to predict match outcomes (win/loss probabilities), forecast cumulative runs, and analyze player performance based on historical IPL data, providing actionable insights for users.
* **Custom Reporting:** Tailored predictions and forecasts for different user needs, including fans, analysts, and strategists, with interactive features to explore match data and player statistics.

**Lessons Learned:**

* **Early Integration Testing Prevents Downstream Challenges**: Ensuring that the machine learning model, data pipeline, and Streamlit app are properly integrated early on helped prevent potential issues during deployment and improved the overall user experience.
* **User Feedback is Invaluable:** Gathering feedback from potential users, such as cricket fans and analysts, provided valuable insights that improved the app’s usability, visualizations, and prediction accuracy.
* **Model Training and Evaluation Are Key to Prediction Accuracy:** Continuous evaluation of different machine learning models helped identify the most accurate models for predicting match outcomes and cumulative runs. Experimentation with features and hyperparameters led to better results.

**Future Enhancements:**

* **Integrating Advanced Predictive Analytics:** Adding more sophisticated models (e.g., Deep Learning, Time Series Forecasting) to improve prediction accuracy and handle more complex factors such as player form, weather, and team dynamics.
* **Adding Real-Time Data Updates:** Incorporating live match data (e.g., player performance, match conditions) for real-time predictions and analysis, allowing the app to adapt during ongoing IPL seasons.
* **Enhancing User Experience**: Improving the app’s interactive features and adding personalized insights based on user preferences, such as predicting the outcome of specific matches or providing detailed player comparisons.

**Appendices**

1. **Sample Code Snippets:**

* **Winner\_prediction\_app**:

import streamlit as st

import pickle

import numpy as np

import pandas as pd

def main():

# Set page configuration

st.set\_page\_config(page\_title="IPL Winning Prediction", page\_icon="📈", layout="wide")

# Sidebar Header

st.sidebar.header("Cricket Match Prediction")

st.sidebar.image("https://upload.wikimedia.org/wikipedia/en/4/41/IPL\_2022\_Logo.png", use\_column\_width=True) # Replace with a relevant IPL image URL

# Main Title and Description

st.title("🏏 IPL Winning Prediction App")

st.markdown("""

This app predicts the \*\*winning probability\*\* of an IPL match based on the match stats.

Enter the details below to get the predictions!

""")

# Load the prediction model

with open("util/ipl\_pred\_model.pkl", "rb") as model\_file:

model = pickle.load(model\_file)

# Define teams and venues

teams = ['MI', 'KKR', 'SRH', 'DC', 'CSK', 'KXIP', 'RR', 'LSG', 'RCB', 'GT'] # Example team names

venues = ['Punjab Cricket Association IS Bindra Stadium, Mohali, Chandigarh',

'M.Chinnaswamy Stadium, Bengaluru',

'Himachal Pradesh Cricket Association Stadium, Dharamsala',

'Arun Jaitley Stadium, Delhi',

'Maharashtra Cricket Association Stadium, Pune',

'Dr DY Patil Sports Academy, Mumbai', 'Eden Gardens, Kolkata',

'Wankhede Stadium, Mumbai', 'Barabati Stadium, Cuttack',

'Holkar Cricket Stadium, Indore', 'Sharjah Cricket Stadium',

'Sawai Mansingh Stadium, Jaipur',

'Sardar Patel (Gujarat) Stadium, Motera, Ahmedabad',

'MA Chidambaram Stadium, Chepauk, Chennai',

'Vidarbha Cricket Association Stadium, Jamtha, Nagpur',

'Sheikh Zayed Stadium, Abu Dhabi',

'Bharat Ratna Shri Atal Bihari Vajpayee Ekana Cricket Stadium, Lucknow',

'SuperSport Park, Centurion',

'Dubai International Cricket Stadium',

'Narendra Modi Stadium, Motera, Ahmedabad',

'Dr DY Patil Sports Academy, Navi Mumbai', 'Newlands, Cape Town',

'Rajiv Gandhi International Stadium, Uppal, Hyderabad',

'JSCA International Stadium Complex, Ranchi',

'Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium, Visakhapatnam',

"St George's Park, Port Elizabeth", 'Brabourne Stadium, Mumbai',

'Kingsmead, Durban',

'Shaheed Veer Narayan Singh International Stadium, Raipur',

'Mangaung Oval, Bloemfontein',

'The Wanderers Stadium, Johannesburg', 'Buffalo Park, East London',

'Diamond Oval, Kimberley', 'Barsapara Cricket Stadium, Guwahati']

# Collect user inputs

st.sidebar.subheader("Match Details")

batting\_team = st.sidebar.selectbox("Batting Team", teams)

bowling\_team = st.sidebar.selectbox("Bowling Team", teams)

toss\_winner = st.sidebar.selectbox("Toss Winner", teams)

venue = st.sidebar.selectbox("Venue", venues)

runs\_scored = st.number\_input("🏏 Runs Scored", min\_value=0, max\_value=500, value=0, step=1)

wickets\_down = st.number\_input("⚾ Wickets Down", min\_value=0, max\_value=10, value=0, step=1)

current\_over = st.number\_input("Over (Completed)", min\_value=0, max\_value=20, value=0, step=1)

current\_ball = st.number\_input("Balls in Current Over", min\_value=0, max\_value=6, value=0, step=1)

target\_runs = st.number\_input("🎯 Target Runs", min\_value=0, max\_value=500, value=0, step=1)

# Calculate derived inputs

runs\_left = target\_runs - runs\_scored

balls\_left = 20 \* 6 - (current\_over \* 6 + current\_ball)

wickets\_remaining = 10 - wickets\_down

crr = round((runs\_scored \* 6) / (current\_over \* 6 + current\_ball), 2) if (current\_over \* 6 + current\_ball) != 0 else 0

rrr = round((runs\_left \* 6) / balls\_left, 2) if balls\_left > 0 else 0

# Prepare input DataFrame

input\_data = pd.DataFrame({

'BattingTeam': [batting\_team],

'BowlingTeam': [bowling\_team],

'runs\_left': [runs\_left],

'balls\_left': [balls\_left],

'wickets\_remaining': [wickets\_remaining],

'target\_runs': [target\_runs],

'crr': [crr],

'rrr': [rrr],

'toss\_winner': [toss\_winner],

'venue': [venue]

})

# Prediction and visualization

if st.button("💡 Predict Winning Probability"):

predicted\_outcome = model.predict(input\_data)[0]

win\_probability = model.predict\_proba(input\_data)[0][1] \* 100

# Display probabilities

st.subheader("🏆 Prediction Results")

col1, col2 = st.columns(2)

with col1:

st.metric(label=f"{batting\_team} Winning Probability", value=f"{round(win\_probability, 2)}%")

with col2:

st.metric(label=f"{bowling\_team} Winning Probability", value=f"{round(100 - win\_probability, 2)}%")

# Progress Bar

st.progress(int(win\_probability))

# Highlight the predicted winner

st.success(f"The most probable winner is: \*\*{predicted\_outcome}\*\*")

if \_\_name\_\_ == '\_\_main\_\_':

main()

* **Score\_prediction\_app:**

import streamlit as st

import pickle

import pandas as pd

# Load the saved XGBoost model

with open(r"util\xgb\_model.pkl", 'rb') as f:

model = pickle.load(f)

# Load the LabelEncoder for team encoding

with open(r"util\team\_encoder.pkl", 'rb') as f:

team\_encoder = pickle.load(f)

# Apply custom CSS for better styling

st.markdown("""

<style>

.main {

background-color: black;

font-family: Arial, sans-serif;

}

.sidebar .sidebar-content {

background-color: #0066cc;

color: white;

}

.stButton button {

background-color: #28a745;

color: white;

font-size: 16px;

}

.stSlider > div {

color: #0066cc;

}

</style>

""", unsafe\_allow\_html=True)

# App title and description

st.title("📊 IPL Run Prediction App")

st.subheader("🏏 Predict the cumulative runs of an ongoing match based on the match's features.")

# Sidebar for user input

st.sidebar.header("🎯 Input Match Features")

st.sidebar.markdown("Provide the details of the match to predict the cumulative runs:")

# Dropdowns for team selection

teams = list(team\_encoder.classes\_)

home\_team = st.sidebar.selectbox("🏠 Select Home Team", teams)

away\_team = st.sidebar.selectbox("🚩 Select Away Team", teams)

current\_innings = st.sidebar.selectbox("⚾ Current Batting Team", teams)

# Sliders and input for match data

over = st.sidebar.slider("Over", 0, 20, 10)

ball = st.sidebar.slider("Ball", 0, 6, 3)

run\_rate = st.sidebar.number\_input("📈 Current Run Rate", min\_value=0.0, max\_value=20.0, value=6.0, step=0.1)

wickets\_lost = st.sidebar.slider("⚡ Wickets Lost", 0, 10, 2)

# Prediction logic

if st.sidebar.button("🔍 Predict"):

# Encode team inputs

home\_team\_encoded = team\_encoder.transform([home\_team])[0]

away\_team\_encoded = team\_encoder.transform([away\_team])[0]

current\_innings\_encoded = team\_encoder.transform([current\_innings])[0]

# Create input DataFrame

input\_data = pd.DataFrame({

'over': [over],

'ball': [ball],

'run\_rate': [run\_rate],

'wickets\_lost': [wickets\_lost],

'home\_team\_encoded': [home\_team\_encoded],

'away\_team\_encoded': [away\_team\_encoded],

'current\_innings\_encoded': [current\_innings\_encoded]

})

# Make prediction

prediction = model.predict(input\_data)

# Display prediction

st.subheader("📊 Prediction Result")

st.success(f"🏏 \*\*Predicted Cumulative Runs:\*\* {prediction[0]:.0f}")

st.markdown(

"The predicted cumulative runs are based on the current match scenario. Use this prediction for strategic decision-making!"

)

* **Forecasting\_app:**

import streamlit as st

import pandas as pd

from PIL import Image

import plotly.graph\_objects as go

import util.\_forecasting as \_forecasting

# Streamlit app configuration

st.set\_page\_config(

page\_title="IPL Team Points Forecasting",

page\_icon="🌟",

layout="centered",

initial\_sidebar\_state="expanded",

)

# Title and description

st.title("🏏 IPL Team Points Forecasting App")

st.write(

"""

Predict how your favorite IPL team will perform in the upcoming seasons!

Select a team, choose the number of future seasons to forecast, and view the results in an interactive chart.

"""

)

# Dataset path

DATASET\_PATH = "util/processed\_points.csv"

# Team selection dropdown

teams = [

"Gujarat Titans", "Chennai Super Kings", "Lucknow Super Giants",

"Mumbai Indians", "Rajasthan Royals",

"Royal Challengers Bangalore", "Kolkata Knight Riders",

"Kings XI Punjab", "Delhi Capitals", "Sunrisers Hyderabad"

]

st.sidebar.header("Forecast Parameters")

team\_name = st.sidebar.selectbox(

"Select a Team:",

options=teams,

help="Choose the IPL team for which you want to forecast points."

)

# Number of seasons input

steps = st.sidebar.number\_input(

"Forecast Seasons:",

min\_value=3,

max\_value=10,

value=5,

step=1,

help="Enter the number of future seasons to forecast (between 3 and 10)."

)

# Action button

if st.sidebar.button("🌟 Forecast"):

if team\_name and steps > 0:

try:

# Call the forecasting function

forecast\_results, future\_seasons, historical\_data = \_forecasting.forecast\_points(team\_name, DATASET\_PATH, steps)

if isinstance(forecast\_results, dict):

# Display forecast results

st.subheader(f"📊 Forecast Results for {team\_name}")

forecast\_df = pd.DataFrame(list(forecast\_results.items()), columns=["Season", "Predicted Points"])

st.dataframe(forecast\_df, use\_container\_width=True)

# Plot forecast results interactively using Plotly

st.subheader(f"📈 Forecasting Plot for {team\_name}")

fig = go.Figure()

# Add historical data

fig.add\_trace(go.Scatter(

x=historical\_data["Season"],

y=historical\_data["Match Points"],

mode='lines+markers',

name='Historical Data',

line=dict(color='blue'),

marker=dict(size=8)

))

# Add forecasted data

fig.add\_trace(go.Scatter(

x=future\_seasons,

y=list(forecast\_results.values()),

mode='lines+markers',

name='Forecasted Data',

line=dict(dash='dash', color='green'),

marker=dict(size=8)

))

# Customize layout

fig.update\_layout(

title=f"{team\_name} Points Forecast",

xaxis\_title="Season",

yaxis\_title="Match Points",

legend\_title="Data Type",

template="plotly\_white",

hovermode="x unified"

)

st.plotly\_chart(fig, use\_container\_width=True)

else:

st.error(f"An error occurred during forecasting: {forecast\_results}")

except Exception as e:

st.error(f"An unexpected error occurred: {str(e)}")

else:

st.error("Please select a team and enter a valid number of forecast seasons.")

# Footer

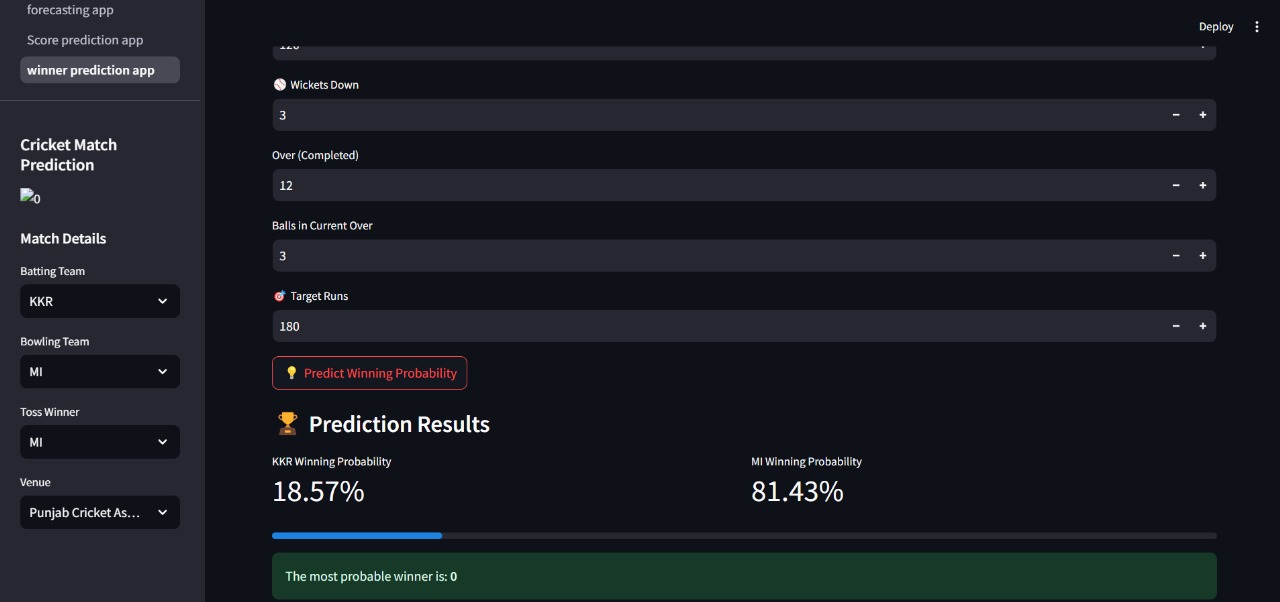
st.write("---")

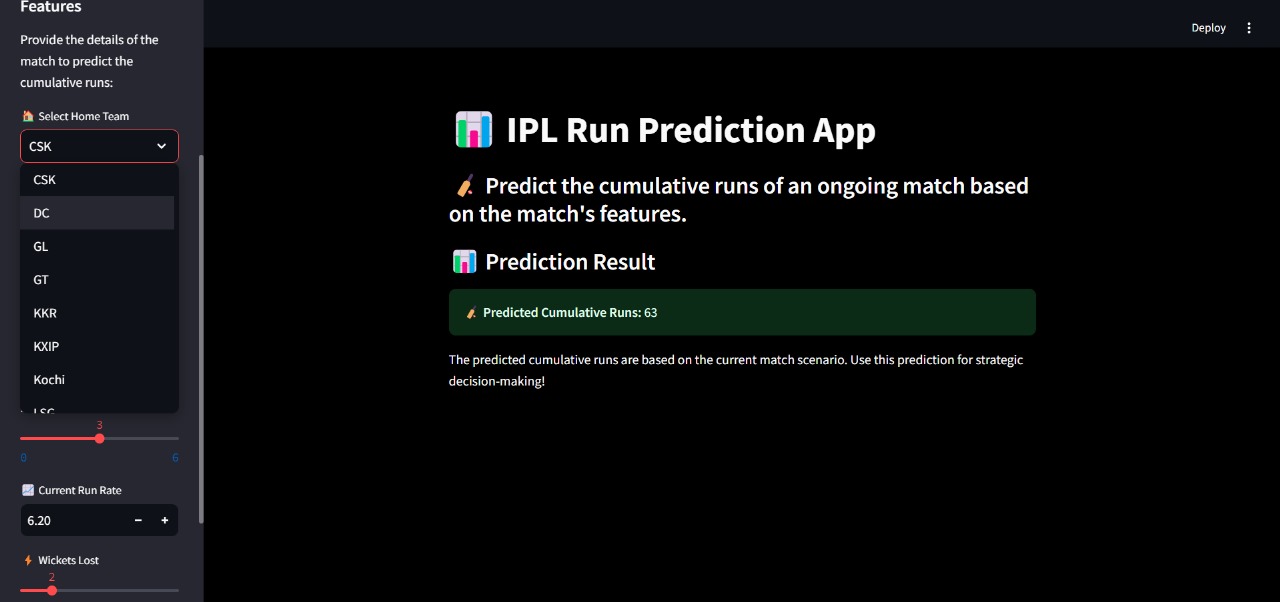
st.write(

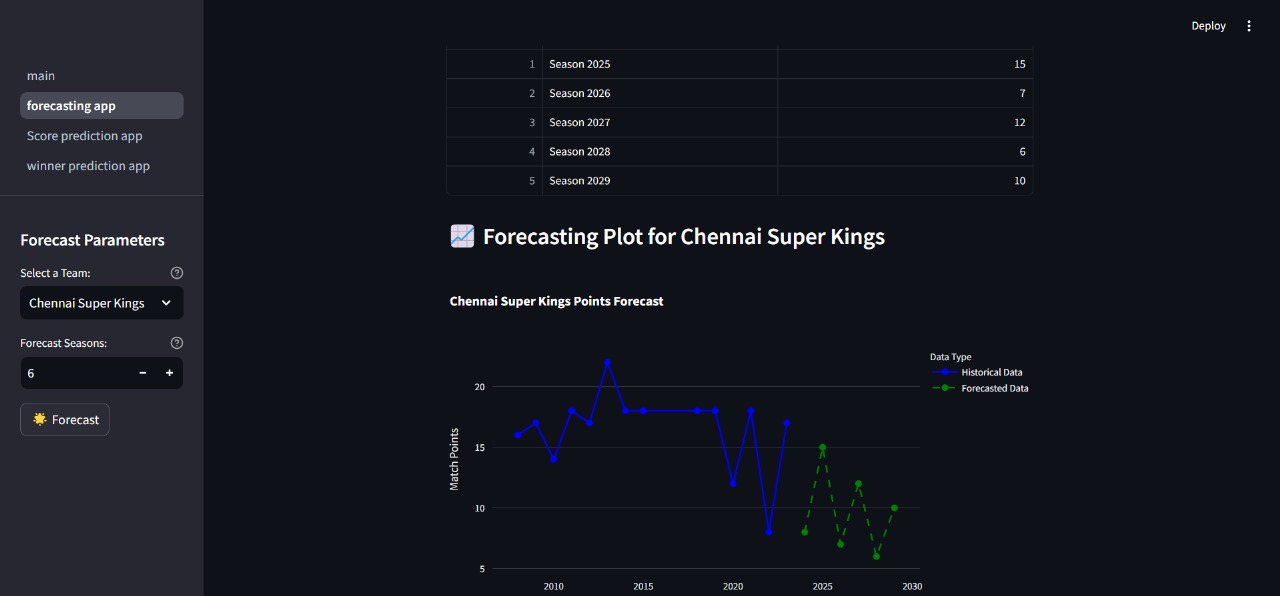
"Developed with ❤️ for IPL enthusiasts. Have fun forecasting!"

)

**2.Web Application:**







**3.References:** All\_season\_details Dataset, scikit-learn

website(https://scikit-learn.org), pandas tutorials,Matplotlib Tutorial.