IPL WINNER PREDICTION USING MACHINE LEARNING

Submitted by

Sai Venkata Satya Krishna Mukku

21EM1A05A5

SWARNANDHRA INSTITUTE OF ENGINEERING AND TECHNOLOGY

NARSAPUR , 534275

TABLE OF CONTENTS

ABSTRACT:

In the landscape of modern cricket, the Indian Premier League (IPL) stands out as a dynamic and fiercely competitive tournament. This project delves into the realm of sports analytics by leveraging machine learning (ML) techniques to forecast the outcomes of IPL matches. By analyzing extensive historical data encompassing player statistics, match venues, team compositions, and past performance metrics, this study aims to develop predictive models that enhance understanding and anticipation of match results.

Summary of the Project:

The project begins with meticulous data collection from reliable sources, ensuring a comprehensive dataset that captures the nuances of IPL matches over multiple seasons. Data preprocessing techniques are applied to cleanse and transform the raw data, addressing issues such as missing values and categorical variables. Feature engineering plays a pivotal role in extracting meaningful insights, creating new metrics that encapsulate team strengths, player form, and situational dynamics.

Several ML algorithms are evaluated to determine their efficacy in predicting IPL match winners. Decision trees, random forests, support vector machines (SVM), and logistic regression are among the models explored, each offering unique capabilities in handling the complexities inherent in cricket match predictions. Model evaluation employs robust metrics such as accuracy, precision, recall, and F1-score to assess performance and identify the most reliable predictors of match outcomes.

Key Findings and Contributions:

Through rigorous analysis and experimentation, the study identifies critical factors influencing IPL match results. Key findings highlight the significance of player performance metrics, the impact of match venue and conditions, the influence of team composition, and the pivotal role of toss outcomes. These insights not only enhance our understanding of the game but also provide actionable intelligence for cricket enthusiasts, team strategists, and stakeholders in the sports industry.

Contributions from this research extend beyond predictive modeling, fostering advancements in sports analytics and decision-making processes. The findings underscore the potential of ML in refining predictions and optimizing strategies, paving the way for future innovations in sports management and performance analysis.

This project serves as a foundational step towards leveraging data-driven approaches to unravel the complexities of IPL cricket, offering valuable insights that can drive informed decisions and enhance the spectator experience.

1. INTRODUCTION

1.1 Background

- The Indian Premier League (IPL) has revolutionized cricket with its format, combining sporting excellence with entertainment. Since its inception in 2008, IPL has attracted a global audience and become a breeding ground for talent and innovation in cricket.
- The unpredictable nature of T20 cricket, coupled with the diverse dynamics of team compositions and match conditions, presents a unique challenge and opportunity for predictive modeling .

1.2 Objectives

- The primary objective of this study is to develop robust machine learning models that can accurately predict the winners of IPL matches.
- Secondary objectives include identifying key factors influencing match outcomes, evaluating the performance of different ML algorithms, and exploring the impact of variables such as player statistics, match venues, and team dynamics on predictions.

1.3 Scope of the Study

- This study focuses on historical IPL match data spanning multiple seasons, encompassing player performances, team compositions, match venues, toss outcomes, and other relevant variables.
- The scope includes data collection, pre processing, feature engineering, model training, evaluation, and interpretation, aiming to provide comprehensive insights into the predictive capabilities of ML in the context of IPL.

1.4 Methodology Overview

- **Data Collection:** Comprehensive gathering of IPL match data from reliable sources, ensuring completeness and accuracy.
- **Data Pre processing**: Cleaning and preparing the dataset to handle missing values, outliers, and categorical variables.
- **Feature Engineering**: Creating new features and transforming existing ones to enhance the predictive power of the models.
- **Model Selection**: Evaluating and selecting suitable ML algorithms (such as decision trees, random forests, SVM, and logistic regression) based on their performance metrics.
- **Model Training and Evaluation** : Training models on historical data and evaluating their performance using metrics like accuracy, precision, recall, and F1-score.

2. Project Overview

2.1 Data Collection

- **Sources**: Gathering IPL match data from reliable sources such as official IPL websites, cricket statistics databases, and reputable sports analytics platforms.
- **Data Variables**: Collecting variables including match outcomes, player statistics (batting average, bowling economy, strike rates), team compositions, match venues, toss outcomes, and other relevant factors.
- **Data Integrity**: Ensuring data integrity through validation and verification processes to maintain accuracy and completeness.

2.2 Data Preprocessing

- **Cleaning**: Handling missing data, duplicates, and outliers through techniques like imputation, removal, or interpolation.
- **Normalization**: Scaling numerical features to a uniform range to prevent any single feature from dominating the model.
- **Encoding**: Converting categorical variables into numerical representations suitable for machine learning algorithms (e.g., one-hot encoding).

2.3 Feature Engineering

- **Creation of Features**: Generating new features that capture unique insights, such as player performance averages, team strength indicators, historical performance trends, and venue-specific metrics.
- **Transformation**: Applying transformations like logarithmic transformations or polynomial features to enhance the predictive power of the models.
- **Selection**: Selecting the most relevant features using techniques such as correlation analysis or feature importance rankings.

2.4 Exploratory Data Analysis (EDA)

- **Visualization**: Creating visual representations (plots, charts, graphs) to explore relationships between variables, identify patterns, and uncover anomalies.
- **Statistical Analysis**: Computing summary statistics and conducting hypothesis testing to gain deeper insights into the dataset.
- **Insights**: Deriving actionable insights that inform subsequent modeling decisions and highlight potential predictors of IPL match outcomes.

2.5 Tools and Technologies Used

- **Programming Languages**: Utilizing languages such as Python or R for data manipulation, statistical analysis, and machine learning model implementation.
- **Libraries**: Leveraging libraries like pandas, NumPy, scikit-learn, and matplotlib for data processing, modeling, and visualization.
- **Development Environment**: Working within environments such as Jupyter Notebooks or integrated development environments (IDEs) for code development and experimentation.

3. Algorithms Used

3.1 Decision Trees

- **Description**: Decision trees partition the data into subsets based on features, making sequential decisions to predict outcomes.
- **Advantages**: Intuitive visualization, ability to handle both numerical and categorical data.
- **Applications**: Used for their interpretability and ability to capture non-linear relationships in IPL match data.

3.2 Random Forest

- **Description**: Ensemble method consisting of multiple decision trees, each trained on a subset of the data and features.
- **Advantages**: Reduced risk of over fitting, improved accuracy and robustness compared to individual decision trees.
- **Applications**: Effective for predicting IPL match winners by aggregating predictions from multiple trees.

3.3 Support Vector Machines (SVM)

- **Description**: SVMs find the optimal hyper plane that separates data points into different classes, maximizing the margin between classes.
- **Advantages**: Effective in high-dimensional spaces, versatile due to different kernel functions.
- **Applications**: Applied to classify IPL match outcomes based on historical data features.

3.4 Logistic Regression

- **Description**: Regression model used for binary classification tasks, predicting the probability of a match outcome.
- **Advantages**: Simple yet powerful, interpretable results in terms of probability.
- **Applications**: Predicting IPL match winners by estimating probabilities based on historical data features.

3.5 Model Selection Criteria

- **Metrics**: Evaluating models based on metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC).
- **Cross-Validation**: Implementing techniques like k-fold cross-validation to ensure robustness and generalizability of model performance.
- **Selection Process**: Selecting the best-performing model based on validation results and tuning parameters for optimal performance in predicting IPL winners.

4. SOURCE CODE

```
import numpy as np
import pandas as pd
```

```
import os
for dirname,_,flesnames in os.walk('Kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname,filename))
```

```
pd.set_option('display.max_columns',None)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import ExtraTreesClassifier,RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,roc_auc_score,classification_report
```

```
mdf=pd.read_csv('matches.csv')
mdf.shape
```

```
mdf_clone=mdf.copy()
```

```
pd.concat([mdf_clone['team1'], mdf_clone['team2']]).unique()
```

```
team1_win_count = len(mdf.loc[mdf['team1'] == mdf['winner']])
team2_win_count = len(mdf.loc[mdf['team2'] == mdf['winner']])
team1_win_count, team2_win_count
```

```
mdf['dayofyear'] = pd.to_datetime(mdf['date']).dt.dayofyear
mdf['dayofweek'] = pd.to_datetime(mdf['date']).dt.dayofweek
mdf['season'] = pd.to_datetime(mdf['date']).dt.year
```

```
mdf = mdf.drop(columns=['date'])
```

```
team_city_map = {
    'Sunrisers Hyderabad': 'Hyderabad',
    'Deccan Chargers': 'Hyderabad',
    'Mumbai Indians': 'Mumbai',
    'Gujarat Lions': 'Gujarat',
    'Gujarat Titans': 'Gujarat',
    'Rising Pune Supergiants': 'Pune',
    'Rising Pune Supergiant': 'Pune',
    'Pune Warriors': 'Pune',
    'Royal Challengers Bangalore': 'Bangalore',
    'Royal Challengers Bengaluru': 'Bangalore',
    'Kolkata Knight Riders': 'Kolkata',
    'Delhi Daredevils': 'Delhi',
    'Delhi Capitals': 'Delhi',
    'Kings XI Punjab': 'Punjab',
    'Punjab Kings': 'Punjab',
    'Chennai Super Kings': 'Chennai',
    'Rajasthan Royals': 'Rajasthan',
    'Kochi Tuskers Kerala': 'Kochi',
    'Lucknow Super Giants': 'Lucknow'
}


def team_to_city(df, feature):
        df[feature] = df[feature].map(team_city_map)
    return df


def map_team_city(df):
    mdf = df.copy()
    for feature in ['team1', 'team2', 'toss_winner', 'winner']:
        if feature in mdf.columns:
            mdf = team_to_city(mdf, feature)
    return mdf


# Apply the function to the DataFrame
mdf = map_team_city(mdf)
```

```
mdf['team1_home'] = False
mdf['team2_home'] = False


mdf.loc[mdf['city'] == mdf['team1'], 'team1_home'] = True
mdf.loc[mdf['city'] == mdf['team2'], 'team2_home'] = True
```

```
mdf = mdf.drop(mdf.loc[mdf['result'] == 'no result'].index, axis=0)
```

```
mdf.head()
```

```
mdf['target_overs'].unique()
```

```
mdf.isna().sum()
```

```
mdf = mdf.drop(columns=['method'])
mdf = mdf.dropna()
```

```
import math
total_teams_score = {}
total_teams_balls = {}0
match_tosswinner_count = {}
match_innings2_team_count = {}
avg_teams_score = {}
avg_teams_balls_left = {}

def team_avg_scores(df):
    df = df.copy()
    for index, row in df.iterrows():
        toss_winner = row['toss_winner']
        target_runs = row['target_runs']
        target_overs = row['target_overs']
        innings2_team = row['team2'] if row['team1'] == row['toss_winner'] else row['team1']

        if toss_winner not in total_teams_score:
            total_teams_score[toss_winner] = 0
        if toss_winner not in match_tosswinner_count:
            match_tosswinner_count[toss_winner] = 0
        if innings2_team not in total_teams_balls:
            total_teams_balls[innings2_team] = 0
        if innings2_team not in match_innings2_team_count:
            match_innings2_team_count[innings2_team] = 0

        total_teams_score[toss_winner] += target_runs
```

```
            num_balls = (math.floor(target_overs) - 1) * 6 + (target_overs -
math.floor(target_overs)) * 10
        total_teams_balls[innings2_team] += (120 - num_balls)
        # 20 overs is 120 balls. We calculate how many balls they were left with.
        match_tosswinner_count[toss_winner] += 1
        match_innings2_team_count[innings2_team] += 1

    # Calculate the average scores and overs
    for team in total_teams_score.keys():
        avg_teams_score[team] = total_teams_score[team] / match_tosswinner_count[team]
    for team in total_teams_balls.keys():
        avg_teams_balls_left[team] = total_teams_balls[team] /
match_innings2_team_count[team]

    # Add new features to the DataFrame
    df['team1_avg_score'] = df['team1'].apply(lambda x: avg_teams_score.get(x, 0))
    df['team2_avg_score'] = df['team2'].apply(lambda x: avg_teams_score.get(x, 0))
    df['team1_avg_balls_left'] = df['team1'].apply(lambda x: avg_teams_balls_left.get(x, 0))
    df['team2_avg_balls_left'] = df['team2'].apply(lambda x: avg_teams_balls_left.get(x, 0))

    return df

# Apply the function to the DataFrame
mdf = team_avg_scores(mdf)
```

```
mdf.head()
```

```
total_teams_balls
```

```
mdf.head()
```

```
mdf['city'] = mdf['city'].fillna('Mumbai')
mdf['umpire1'] = mdf['umpire1'].fillna('HDPK Dharmasena')
mdf['umpire2'] = mdf['umpire2'].fillna('S Ravi')
```

```
mdf = pd.get_dummies(mdf, columns = ['toss_decision'], drop_first=True)
```

```
mdf = mdf.drop(columns=['id', 'result'])
```

```
mdf.head()
```

```
mdf = mdf.drop(columns=['player_of_match'])
```

```
matches_won = dict(mdf['winner'].value_counts())
```

```
team1_played = dict(mdf['team1'].value_counts())
team2_played = dict(mdf['team2'].value_counts())

matches_played = {}
for team, count in team1_played.items():
    if team not in matches_played.keys():
        matches_played[team] = 0
    matches_played[team] += count

for team, count in team2_played.items():
    if team not in matches_played.keys():
        matches_played[team] = 0
    matches_played[team] += count
```

```
win_ratio = {}
```

```
for team, count in matches_played.items():
    win_ratio[team] = matches_won[team] / count

win_ratio
```

```
mdf.head()
```

```
mdf['team1_win_ratio'] = mdf['team1'].map(win_ratio)
mdf['team2_win_ratio'] = mdf['team2'].map(win_ratio)
```

```
team_le = LabelEncoder()
team_name_features = pd.concat([mdf['city'], mdf['team1'], mdf['team2'], mdf['winner'],
mdf['toss_winner']]).unique()
team_le.fit(team_name_features)

mdf['city'] = team_le.transform(mdf['city'])
mdf['winner'] = team_le.transform(mdf['winner'])
mdf['team1'] = team_le.transform(mdf['team1'])
mdf['team2'] = team_le.transform(mdf['team2'])
mdf['toss_winner'] = team_le.transform(mdf['toss_winner'])
```

```
le = LabelEncoder() # Use a different instance of the label encoder for the other features.

features = ['venue', 'umpire1', 'umpire2']
venue_le = LabelEncoder()
umpire_le = LabelEncoder()
umpire_le.fit(pd.concat([mdf['umpire1'], mdf['umpire2']]).unique())
mdf['venue'] = venue_le.fit_transform(mdf['venue'])
mdf['umpire1'] = umpire_le.transform(mdf['umpire1'])
mdf['umpire2'] = umpire_le.transform(mdf['umpire2'])
```

```
mdf.head()
```

```
mdf = mdf.drop(columns=['super_over'])
```

```
mdf.columns
```

```
mdf = mdf.reindex(columns=['season', 'dayofyear', 'dayofweek', 'venue', 'city',
                           'team1', 'team2', 'team1_win_ratio', 'team2_win_ratio', 'team1_home',
'team2_home',
                           'team1_avg_score', 'team2_avg_score', 'team1_avg_balls_left',
'team2_avg_balls_left',
                           'toss_winner',  'toss_decision_field',
                           'umpire1', 'umpire2',
                           'winner'])
```

```
mdf['team1_wins'] = (mdf['winner'] == mdf['team1']).astype(int)
mdf = mdf.drop(columns=['winner'])
mdf.head()
```

```
fig, ax = plt.subplots(figsize=(20, 10))
corr_matrix = mdf.corr()
mask = np.zeros_like(corr_matrix, dtype=np.bool_)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(corr_matrix, cmap=sns.diverging_palette(20, 220, n=200), annot=True,
mask=mask, center = 0)
plt.show()
```

```
X = mdf.drop(columns=['team1_wins'])
```

```
y = mdf['team1_wins']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
models_list = []
num_estimators_list =  [100, 200, 500, 1000, 1500, 2000]
max_depth_list = [3, 4, 5, 6, 7]
count = 0
for n_estimators in num_estimators_list:
    for max_depth in max_depth_list:
        model = ExtraTreesClassifier(max_depth=max_depth, n_estimators=n_estimators)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        ATrS = model.score(X_train, y_train)
        ATeS = model.score(X_test, y_test)
        models_list.append(pd.Series({"n_estimators": n_estimators, "max_depth": max_depth,
"ATrS": ATrS, "ATeS": ATeS}))
        count += 1
        print(f"Finished: {count}/{len(num_estimators_list)*len(max_depth_list)}")
extra_models = pd.DataFrame(models_list)
extra_models.head(30)
```

```
model = ExtraTreesClassifier(n_estimators=500, max_depth=3)
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)
```

```
print('ATrS:', model.score(X_train, y_train))
print('ATeS:', model.score(X_test, y_test))
print(classification_report(y_test, y_pred))
```

```
mdf_clone['venue'].unique()
```

```
avg_teams_score
```

```
def process_test_data(df):
    df = df.copy()

    df['season'] = pd.to_datetime(df['date'], format='%Y-%m-%d').dt.year
    df['dayofyear'] = pd.to_datetime(df['date'], format='%Y-%m-%d').dt.dayofyear
    df['dayofweek'] = pd.to_datetime(df['date'], format='%Y-%m-%d').dt.dayofweek
    df = df.drop(columns=['date'])

#     df = map_team_city(df)

    df['team1_home'] = False
    df['team2_home'] = False

    df.loc[df['city'] == df['team1'], 'team1_home'] = True
    df.loc[df['city'] == df['team2'], 'team2_home'] = True

    df['team1_win_ratio'] = df['team1'].map(win_ratio)
    df['team2_win_ratio'] = df['team2'].map(win_ratio)

    df['team1_avg_score'] = df['team1'].map(avg_teams_score)
    df['team2_avg_score'] = df['team2'].map(avg_teams_score)
    df['team1_avg_balls_left'] = df['team1'].map(avg_teams_balls_left)
    df['team2_avg_balls_left'] = df['team2'].map(avg_teams_balls_left)
```

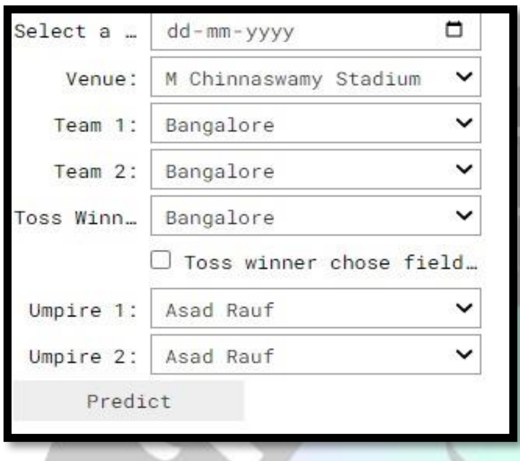
```
        df['team1'] = team_le.transform(df['team1'])
    df['team2'] = team_le.transform(df['team2'])
    df['city'] = team_le.transform(df['city'])
    df['toss_winner'] = team_le.transform(df['toss_winner'])

    df['venue'] = venue_le.transform(df['venue'])
    df['umpire1'] = umpire_le.transform(df['umpire1'])
    df['umpire2'] = umpire_le.transform(df['umpire2'])

    df = df.reindex(columns=['season', 'dayofyear', 'dayofweek', 'venue', 'city',
                            'team1', 'team2', 'team1_win_ratio', 'team2_win_ratio', 'team1_home',
'team2_home',
                            'team1_avg_score', 'team2_avg_score', 'team1_avg_balls_left',
'team2_avg_balls_left',
                            'toss_winner',  'toss_decision_field',
                            'umpire1', 'umpire2'])

    return df
```

```
from IPython.display import display
import ipywidgets as widgets
import pandas as pd
from datetime import datetime

teams = win_ratio.keys()
venues = mdf_clone['venue'].unique()
umpires = pd.concat([mdf_clone['umpire1'], mdf_clone['umpire2']]).unique()

stadium_city_map = {
      'M Chinnaswamy Stadium': 'Bangalore',
          'M Chinnaswamy Stadium, Bengaluru': 'Bangalore',
      'Punjab Cricket Association Stadium, Mohali': 'Mohali',
      'Punjab Cricket Association IS Bindra Stadium, Mohali': 'Mohali',
      'Feroz Shah Kotla': 'Delhi',
      'Arun Jaitley Stadium': 'Delhi',
      'Arun Jaitley Stadium, Delhi': 'Delhi',
      'Wankhede Stadium': 'Mumbai',
      'Wankhede Stadium, Mumbai': 'Mumbai',
      'Eden Gardens': 'Kolkata',
      'Eden Gardens, Kolkata': 'Kolkata',
      'Sawai Mansingh Stadium': 'Jaipur',
      'Sawai Mansingh Stadium, Jaipur': 'Jaipur',
      'Rajiv Gandhi International Stadium, Uppal': 'Hyderabad',
      'Rajiv Gandhi International Stadium': 'Hyderabad',
      'Rajiv Gandhi International Stadium, Uppal, Hyderabad': 'Hyderabad',
      'MA Chidambaram Stadium, Chepauk': 'Chennai',
      'MA Chidambaram Stadium': 'Chennai',
      'MA Chidambaram Stadium, Chepauk, Chennai': 'Chennai',
      'Dr DY Patil Sports Academy': 'Mumbai',
      'Dr DY Patil Sports Academy, Mumbai': 'Mumbai',
      'Newlands': 'Cape Town',
      "St George's Park": 'Port Elizabeth',
      'Kingsmead': 'Durban',
      'SuperSport Park': 'Centurion',
      'Buffalo Park': 'East London',
      'New Wanderers Stadium': 'Johannesburg',
      'De Beers Diamond Oval': 'Kimberley',
      'OUTsurance Oval': 'Bloemfontein',
      'Brabourne Stadium': 'Mumbai',
      'Brabourne Stadium, Mumbai': 'Mumbai',
          'Sardar Patel Stadium, Motera': 'Ahmedabad',
      'Barabati Stadium': 'Cuttack',
      'Vidarbha Cricket Association Stadium, Jamtha': 'Nagpur',
      'Himachal Pradesh Cricket Association Stadium': 'Dharamsala',
      'Himachal Pradesh Cricket Association Stadium, Dharamsala': 'Dharamsala',
      'Nehru Stadium': 'Kochi',
      'Holkar Cricket Stadium': 'Indore',
      'Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium': 'Visakhapatnam',
      'Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium, Visakhapatnam':
  'Visakhapatnam',
      'Subrata Roy Sahara Stadium': 'Pune',
      'Maharashtra Cricket Association Stadium': 'Pune',
      'Maharashtra Cricket Association Stadium, Pune': 'Pune',
      'Shaheed Veer Narayan Singh International Stadium': 'Raipur',
      'JSCA International Stadium Complex': 'Ranchi',
      'Sheikh Zayed Stadium': 'Abu Dhabi',
      'Sharjah Cricket Stadium': 'Sharjah',
      'Dubai International Cricket Stadium': 'Dubai',
      'Saurashtra Cricket Association Stadium': 'Rajkot',
      'Green Park': 'Kanpur',
      'Narendra Modi Stadium, Ahmedabad': 'Ahmedabad',
      'Zayed Cricket Stadium, Abu Dhabi': 'Abu Dhabi',
      'Eden Gardens, Kolkata': 'Kolkata',
      'Punjab Cricket Association IS Bindra Stadium': 'Mohali',
      'Punjab Cricket Association IS Bindra Stadium, Mohali, Chandigarh': 'Mohali',
      'Bharat Ratna Shri Atal Bihari Vajpayee Ekana Cricket Stadium, Lucknow': 'Lucknow',
      'Barsapara Cricket Stadium, Guwahati': 'Guwahati',
      'Maharaja Yadavindra Singh International Cricket Stadium, Mullanpur': 'Mullanpur'
  }
```

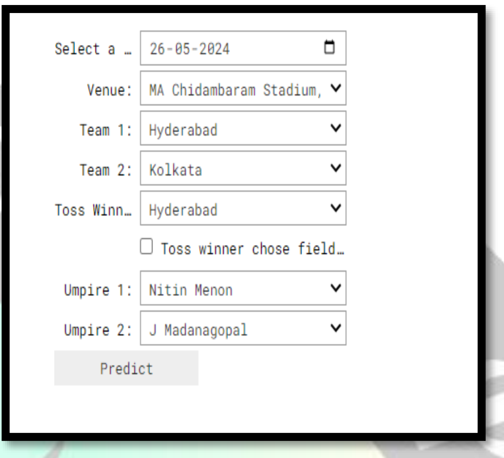
```
date = widgets.DatePicker(description='Select a date', disabled=False)
venue = widgets.Dropdown(options=venues, description='Venue:')
team1 = widgets.Dropdown(options=teams, description='Team 1:')
team2 = widgets.Dropdown(options=teams, description='Team 2:')
toss_winner = widgets.Dropdown(options=teams, description='Toss Winner:')
toss_decision_field = widgets.Checkbox(value=False, description='Toss winner chose
fielding?')
umpire1 = widgets.Dropdown(options=umpires, description='Umpire 1:')
umpire2 = widgets.Dropdown(options=umpires, description='Umpire 2:')

prediction_result = widgets.Output()

def handle_prediction(btn):
    with prediction_result:
        prediction_result.clear_output()
        if team1.value == team2.value:
            print("Team1 and team2 cannot be the same!")
            return
        if umpire1.value == umpire2.value:
            print("One person can't be both umpires!")
            return

        test_data = {
            'date': date.value,
            'venue': venue.value,
            'city': stadium_city_map[venue.value],
            'team1': team1.value,
            'team2': team2.value,
            'toss_winner': toss_winner.value,
            'toss_decision_field': toss_decision_field.value,
                'umpire1': umpire1.value,
            'umpire2': umpire2.value
        }

        if None in test_data.values():
            print("Please fill all fields.")
            return

        test_data_df = pd.Series(test_data).to_frame().T
        test_data_df = process_test_data(test_data_df)
        prediction = model.predict_proba(test_data_df)[0]
        print(f"Processed Test Data:\n{test_data_df}")
        print(f"Prediction: \n{team1.value} - {prediction[1]} \n{team2.value} - 
{prediction[0]}")

predict_button = widgets.Button(description='Predict')
predict_button.on_click(handle_prediction)

input_form = widgets.VBox([
    date, venue, team1, team2, toss_winner, toss_decision_field, umpire1, umpire2,
    predict_button, prediction_result
])

display(input_form)
```

5.RESULTS

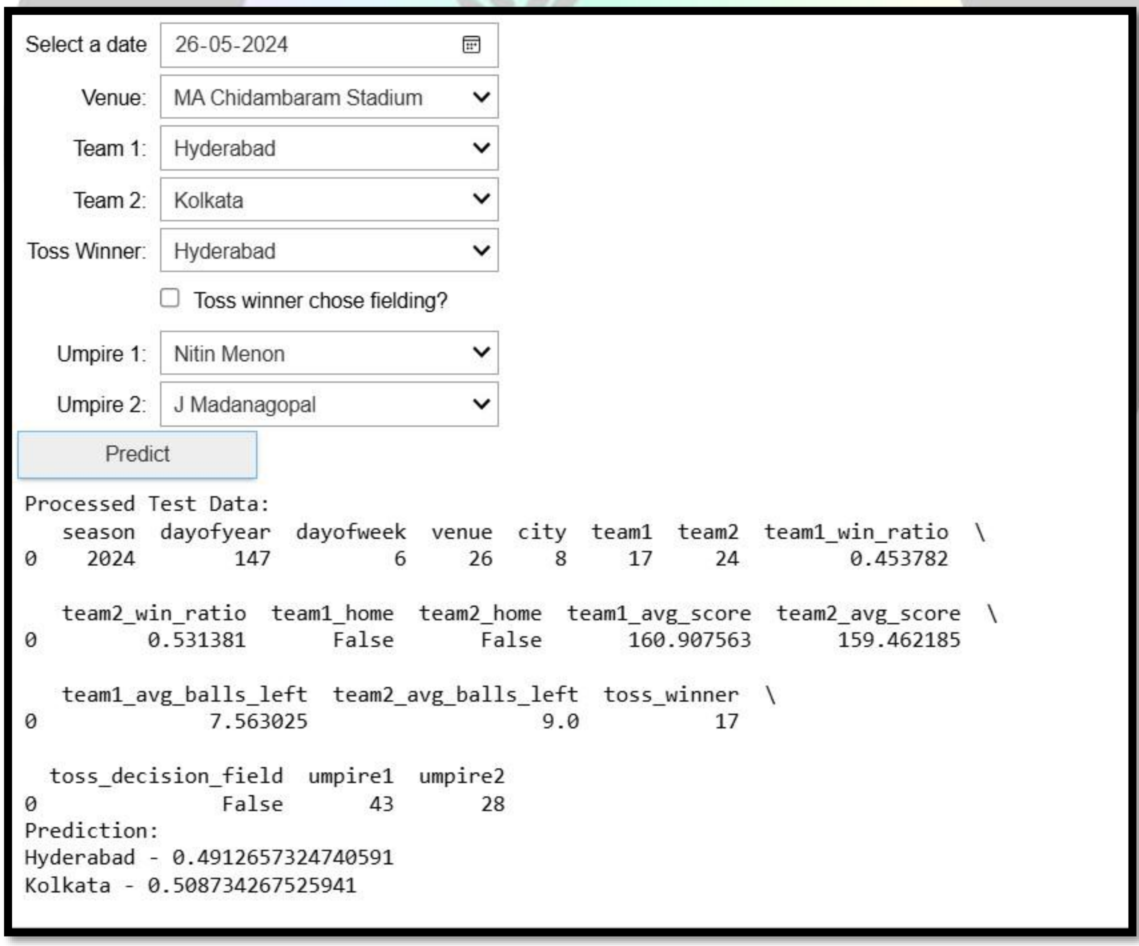
When the above code is executed based on the given datasets we will get the output as shown below:



Select a …	dd-mm-yyyy
Venue:	M Chinnaswamy Stadium
Team 1:	Bangalore
Team 2:	Bangalore
Toss Winn…	Bangalore
	☐ Toss winner chose field…
Umpire 1:	Asad Rauf
Umpire 2:	Asad Rauf

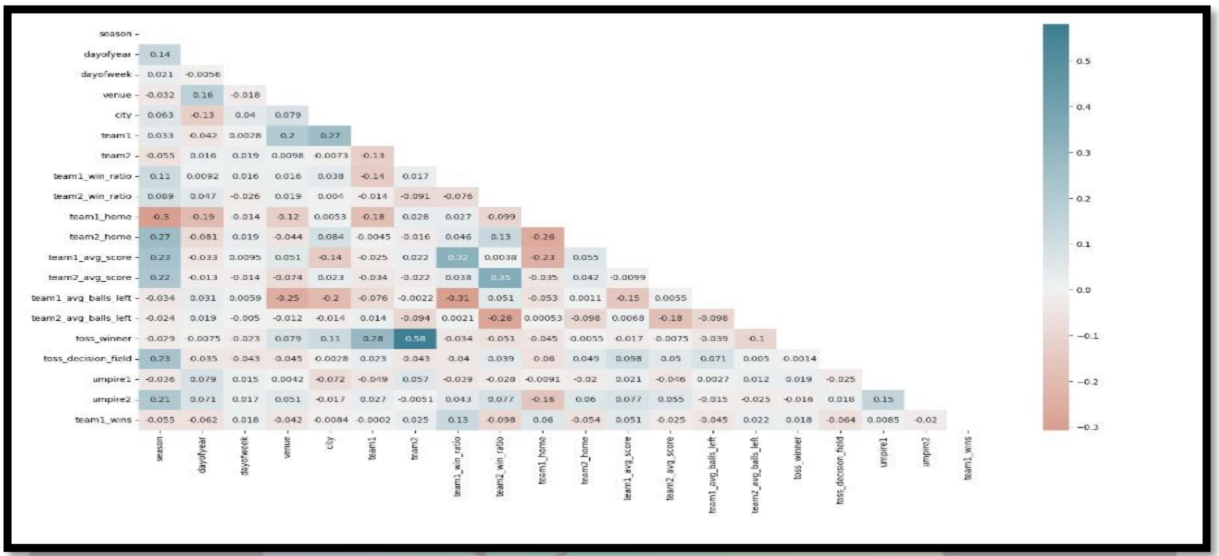


Select a …	26-05-2024
Venue:	MA Chidambaram Stadium,
Team 1:	Hyderabad
Team 2:	Kolkata
Toss Winn…	Hyderabad
	☐ Toss winner chose field…
Umpire 1:	Nitin Menon
Umpire 2:	J Madanagopal



Select a date	26-05-2024
Venue:	MA Chidambaram Stadium
Team 1:	Hyderabad
Team 2:	Kolkata
Toss Winner:	Hyderabad
	☐ Toss winner chose fielding?
Umpire 1:	Nitin Menon
Umpire 2:	J Madanagopal

```
Processed Test Data:
   season  dayofyear  dayofweek  venue  city  team1  team2  team1_win_ratio  \
0    2024        147          6     26     8     17     24         0.453782   

   team2_win_ratio  team1_home  team2_home  team1_avg_score  team2_avg_score  \
0         0.531381       False       False       160.907563       159.462185   

   team1_avg_balls_left  team2_avg_balls_left  toss_winner  \
0              7.563025                   9.0           17   

  toss_decision_field  umpire1  umpire2
0               False       43       28
Prediction:
Hyderabad - 0.4912657324740591
Kolkata - 0.508734267525941
```



	season	dayofyear	dayofweek	venue	city	team1	team2	team1_win_ratio	team2_win_ratio	team1_home	team2_home	team1_avg_score	team2_avg_score	team1_avg_balls_left	team2_avg_balls_left	toss_winner	toss_decision_field	umpire1	umpire2	team1_wins
season																				
dayofyear	0.14																			
dayofweek	0.021	-0.0056																		
venue	-0.032	0.16	-0.018																	
city	0.063	-0.13	0.04	0.079																
team1	0.033	-0.042	0.0028	0.2	0.27															
team2	-0.055	0.016	0.019	0.0098	-0.0073	-0.13														
team1_win_ratio	0.11	0.0092	0.016	0.016	0.038	-0.14	0.017													
team2_win_ratio	0.089	0.047	-0.026	0.019	0.004	-0.014	-0.091	-0.076												
team1_home	-0.3	-0.19	-0.014	-0.12	0.0053	-0.18	0.028	0.027	-0.099											
team2_home	0.27	-0.081	0.019	-0.044	0.084	-0.0045	-0.016	0.046	0.13	-0.26										
team1_avg_score	0.23	-0.033	0.0095	0.051	-0.14	-0.025	0.022	0.32	0.0038	-0.23	0.055									
team2_avg_score	0.22	-0.013	-0.014	-0.074	0.023	-0.034	-0.022	0.038	0.35	-0.035	0.042	-0.0099								
team1_avg_balls_left	-0.034	0.031	0.0059	-0.25	-0.2	-0.076	-0.0022	-0.31	0.051	-0.053	0.0011	-0.15	0.0055							
team2_avg_balls_left	-0.024	0.019	-0.005	-0.012	-0.014	0.014	-0.094	0.0021	-0.28	0.00053	-0.098	0.0068	-0.18	-0.098						
toss_winner	-0.029	-0.0075	-0.023	0.079	0.11	0.28	0.58	-0.034	-0.051	-0.045	-0.0055	-0.017	-0.0075	-0.039	-0.1					
toss_decision_field	0.23	-0.035	-0.043	-0.045	-0.0028	0.023	-0.043	0.04	0.039	-0.06	0.049	0.098	0.05	0.071	0.005	-0.0014				
umpire1	-0.036	0.079	0.015	0.0042	-0.072	-0.049	0.057	-0.039	-0.028	-0.0091	-0.02	0.021	-0.046	0.0027	0.012	0.019	-0.025			
umpire2	0.21	0.071	0.017	0.051	-0.017	0.027	-0.0051	0.043	0.077	-0.16	0.06	0.077	0.055	-0.015	-0.025	-0.016	0.018	0.15		
team1_wins	-0.055	-0.062	0.018	-0.042	-0.0084	-0.0002	0.025	0.13	-0.098	0.06	-0.054	0.051	-0.025	-0.045	0.022	0.018	-0.064	0.0085	-0.02	

6. CONCLUSION

6.1 Summary of Findings

- **Key Insights**: Summarize the main findings and outcomes of the project regarding IPL winner prediction using machine learning.
- **Effective Predictors**: Highlight the factors identified as significant predictors of IPL match outcomes, such as player performance metrics, match venue, team composition, and toss outcomes.
- **Performance of Models**: Discuss the performance of different machine learning algorithms (e.g., decision trees, random forest, SVM) in predicting IPL winners.
- **Limitations**: Address any limitations encountered during the project, such as data availability, model accuracy, and generalizability across different IPL seasons.

6.2 Recommendations for Future Work

- **Advanced Techniques**: Explore advanced machine learning techniques (e.g., deep learning, ensemble methods) to further improve prediction accuracy.
- **Real-time Data Integration**: Incorporate real-time data feeds and updates to enhance the timeliness and relevance of predictions.
- **Feature Engineering**: Develop more sophisticated features that capture nuanced aspects of player and team dynamics, potentially including sentiment analysis of player performance.
- **Model Interpretability**: Enhance model interpretability to provide actionable insights for cricket analysts, team managers, and betting platforms.
- **Validation and Testing**: Conduct rigorous validation and testing across multiple IPL seasons and conditions to validate model robustness and reliability.

7. APPENDENCIES

7.1 Detailed Data Schema

- Provide a comprehensive schema or description of the IPL match data structure used in the project.
- Include details on variables, data types, and relationships between different entities in the dataset.

7.2 Additional Visualizations

- Present supplementary visualizations that provide deeper insights into the IPL match data and model performance.
- Include plots, charts, graphs, and tables that illustrate trends, correlations, and patterns identified during the analysis.

7.3 Glossary of Terms

- Define technical terms, abbreviations, and acronyms used throughout the document.
- Ensure clarity and understanding by providing concise explanations of specialized terminology relevant to IPL analytics and machine learning.