IPL WINNER PREDICTION USING MACHINE LEARNING

Submitted by

ANDEY LAKSHMI AMULYA

21A21A0516

SWARNANDHRA COLLEGE OF

ENGINEERING AND TECHNOLOGY

NARSAPUR 534275

TABLE OF CONTENTS

ABSTRACT:

The Indian Premier League (IPL) is one of the most popular and competitive cricket leagues in the world, characterized by dynamic and unpredictable match outcomes. The focus of this project is to develop a machine learning classification model to predict the winners of IPL matches. Leveraging historical data, including team statistics, player performance metrics, match conditions, and venue specifics, the model will employ various classification algorithms such as Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting. Cricket, especially the T-20 format, is a popular and unpredictable sport where the outcome can change dramatically in a single over. Millions of spectators watch the IPL every year, making it a realtime challenge to create a technique that can forecast match outcomes accurately. This project addresses this challenge by proposing a data-driven approach to predict the winning team of an IPL match. The project begins with data collection from various sources, including historical match data, player statistics, and other relevant features. Data preprocessing is then carried out to clean and prepare the data for analysis. Feature engineering is performed to identify the most relevant predictors for match outcomes. This involves analyzing various aspects and features that determine the result of a cricket match, each of which has a weighted impact on the outcome. The proposed model uses a multivariate regression-based approach to measure the team's points in the league. The past performance of each team is analyzed to estimate its probability of winning against specific opponents. Seven key attributes are identified for predicting the winner of an IPL match. These attributes are then used to train multiple machineLearning models, including Random Forest, Decision Trees, K-Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machines (SVM). The process involves training and evaluating these models using cross-validation techniques to assess their performance and mitigate overfitting. Hyperparameter tuning is employed to optimize the models for improved accuracy. The final model is validated on recent IPL seasons to ensure robustness and reliability. The models' performances are evaluated using various classification techniques, with Random Forest and Decision Tree models demonstrating particularly strong results. In terms of the design and flow of the project, it begins with the data collection phase, followed by data preprocessing and feature engineering. The next phase involves training multiple machine learning models using the identified features. Each model is evaluated using cross-validation techniques, and the best-performing models are selected for hyperparameter tuning. The final model is then validated on recent IPL seasons to ensure its accuracy and reliability. The expected output of this project is a machine learning model that can accurately predict the winner of an IPL match. By providing accurate predictions, this project aims to offer valuable insights for teams, analysts, and enthusiasts, enhancing strategic decision-making and engagement with the game. The study highlights the potential of machine learning in sports analytics, demonstrating how advanced data-driven approaches can be applied to forecast outcomes in competitive environments. This project employs various libraries and technologies to achieve its goals. Python is the primary programming language used, along with libraries such as pandas and NumPy for data manipulation, scikit-learn for building and evaluating machine

learning models, and Matplotlib and Seaborn for data visualization. Additionally, techniques such as cross-validation and hyperparameter tuning are employed to ensure the model's performance and accuracy. In conclusion, this project aims to develop a robust machine learning model to predict the winner of IPL matches. By leveraging historical data and employing various machine learning techniques, the project seeks to provide accurate predictions that can be used for strategic decision-making in the world of cricket. The study demonstrates the potential of machine learning in sports analytics, offering valuable insights and enhancing engagement with the game

INTRODUCTION

The main aim is to use Machine learning to develop the computer programs which will be capable of retrieving data and using it for self-learning. The procedure of learningcommences with observations of data, such as instances, direct experience, or training, in order to identify some patterns in statistics and take improved decisions in the future built on the samples that are provided. Machine Learning primarily aims at eliminating the human intervention or assistance by allowing the computers learn automatically and adjust its actions accordingly. The advancement in computing in the recent years, has made it increasingly easy to acquire in-depth information. As a consequence, the fact of having both live and historic data has made Machine Learning quite popular in the fields of sports analytics [1–5]. Sports Analytics is a method of collecting and analyzing historical game information to derive essential knowledge from it, with the aim that it will promote successful decision-making. Machine learning in sports arena, both off-the-field and on-the-field, can be used effectively on different occasions. A team's performance and its outcome against an opponent can be efficiently predicted using the proposed model. This model primarily focuses on the healthy growth and productivity of team owners and other investors in the industry. Here, analysis is done by using certain machine learning classification techniques, like Decision Trees, Logistic Regression, Support Vector Machine, K-Nearest Neighbors and Random Forest. One of the greatest successful football clubs in Portugal, Sport Lisboa e Benfica implements machine learning in information processing techniques for making decisions , demonstrating the importance of machine learning in athletic analysis. The club not only tracks but also evaluates virtually each part of the game, together with their habit of resting, drinking, and practicing. After capturing raw player data, different models are programmed to analyze data to maximize game preparation and create custom training schedules. The data coming from the built models allow players to constantly improve their performance by incorporating machine learning and predictive analytics. Decisions including player substitution, holding a player in the lineup and leaving a player at bench can be made by the team coach depending on the analysis of the facts obtained. The dataset used in this work is collection of different match plays, there are around 675 match details with complete information about the match winner, location toss winner, team names and other important attributes. The matches are from 2003 – 2024. This dataset has helped us achieve our main aim of our project

PROJECT OVERVIEW

The Indian Premier League (IPL) is one of the most popular and competitive cricket leagues globally, attracting millions of fans and extensive media coverage. Predicting the winner of an IPL match involves analyzing various factors, such as team performance, player statistics, match conditions, and historical data. This project aims to utilize machine learning techniques to predict the outcomes of IPL matches, providing insights and potential forecasts for upcoming games.

Objectives:

The primary objectives of this project are as follows:

- **Data Analysis**: To collect and analysehistorical IPL data, identifying key features and trends that influence match outcomes.
- **Model Development**: To develop and train machine learning models capable of predicting the winner of IPL matches based on the analysed data.
- **Model Evaluation**: To evaluate the performance of the developed models using various metrics, ensuring their reliability and accuracy.
- **Insight Generation**: To generate insights and recommendations based on model predictions, which can be valuable for fans, analysts, and stakeholders.

SCOPE OF THE PROJECT:

The scope of this project encompasses several critical phases:

1. **Data Collection**: Gathering historical IPL match data, including team and player statistics, match conditions, and outcomes.
2. **Data Preprocessing**: Cleaning and preprocessing the collected data to ensure its suitability for machine learning models.
3. **Exploratory Data Analysis (EDA)**: Conducting EDA to uncover patterns, correlations, and important features within the data.
4. **Model Building**: Developing various machine learning models, such as Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines (SVM), to predict match outcomes.
5. **Model Evaluation**: Assessing the performance of the models using metrics like accuracy, precision, recall, and F1 score.
6. **Prediction and Analysis**: Using the trained models to predict the outcomes of future IPL matches and analyzing the results.
7. **Deployment**: Implementing the final model in a user-friendly interface or application for practical use.

ALGORITHMS USED:

Random Forest

Overview

Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It is known for its high accuracy, ability to handle large datasets with higher dimensionality, and robustness to overfitting.

Working of Random Forest

1. **Data Sampling**: The algorithm creates multiple subsets of the original dataset through a method called bootstrap sampling.
2. **Tree Construction**: For each subset, a decision tree is constructed. At each node, a random subset of features is selected, and the best split is chosen from this subset.
3. **Voting**: After all trees are constructed, they make predictions on the test data. For classification tasks, each tree votes for a class, and the class with the majority vote is the final prediction. For regression tasks, the mean prediction of all trees is considered.

Advantages

- **High Accuracy**: By aggregating the predictions of multiple trees, Random Forest usually achieves higher accuracy than individual decision trees.
- **Robustness**: It is less likely to overfit compared to a single decision tree.
- **Feature Importance**: Random Forest can compute the importance of each feature, which can be useful for feature selection.

Implementation in the Project

```
from sklearn.ensemble import RandomForestClassifier
```

Decision Tree

Overview

A Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It splits the data into subsets based on the value of input features, creating a tree-like structure where each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (in classification) or a continuous value (in regression).

Working of Decision Tree

1. **Splitting**: The dataset is split into subsets based on the attribute that results in the most significant information gain or the highest reduction in impurity (e.g., Gini impurity or entropy).
2. **Tree Construction**: This process is repeated recursively for each derived subset, creating a hierarchical tree structure.
3. **Prediction**: For predicting the outcome of a new instance, the instance is passed through the tree, and at each node, a decision is made based on the attribute until a leaf node is reached.

Advantages

- **Interpretability**: Decision Trees are easy to interpret and visualize.
- **Handling Non-linear Relationships**: They can handle non-linear relationships between features and the target variable.
- **No Need for Feature Scaling**: They do not require feature scaling or normalization.

Implementation in the Project

```
from sklearn.tree import DecisionTreeClassifier
```

SOURCE CODE

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames 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
```

```
# ddf = pd.read_csv('/kaggle/input/ipl-complete-dataset-20082020/deliveries.csv')
mdf = pd.read_csv('/kaggle/input/ipl-complete-dataset-20082020/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()
```

```
# Making new features team1_avg_score and team2_avg_score by iterating through the dataset and creating a database (here it's just a dictionary)
import math
total_teams_score = {}
total_teams_balls = {}
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()
```

```
# Imputing NaN values using most_frequent strategy (manually)

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

```
# One-Hot encoding toss_decision

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
```

```
# Calculating the win ratio of every team

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

win_ratio
```

```
mdf.head()
```

```
# Adding team1_win_ratio and team2_win_ratio features to the dataset.
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.colum
```

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

# for i in range(len(y_test)):
#    print(f"Match {i + 1}:")
#    print(f"Predicted Winner: {'Team1' if y_pred[i] == 1 else 'Team2'}")
#    print(f"Probability of Team1 winning: {y_pred_proba[i][1]}")
#    print(f"Probability of Team2 winning: {y_pred_proba[i][0]}")
#    print("\n")
```

```
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
```

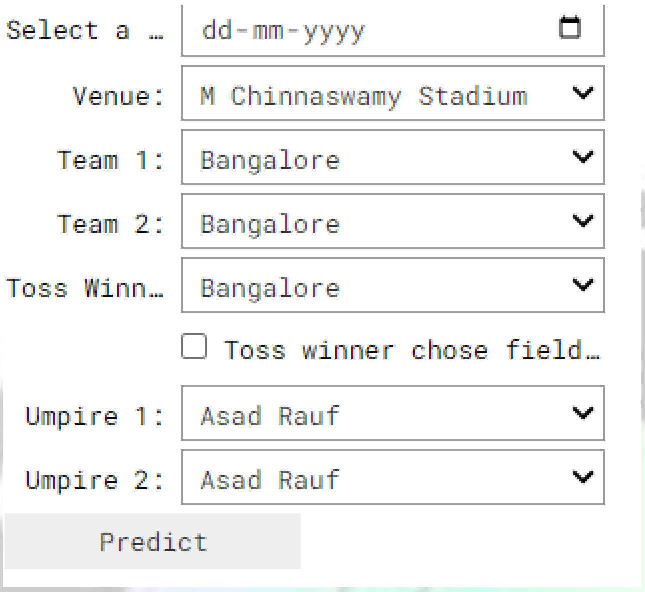
```
from IPython.display import display
import ipywidgets as widgets
import pandas as pd
from datetime import datetime

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)
```

if we execute the above code the output will come

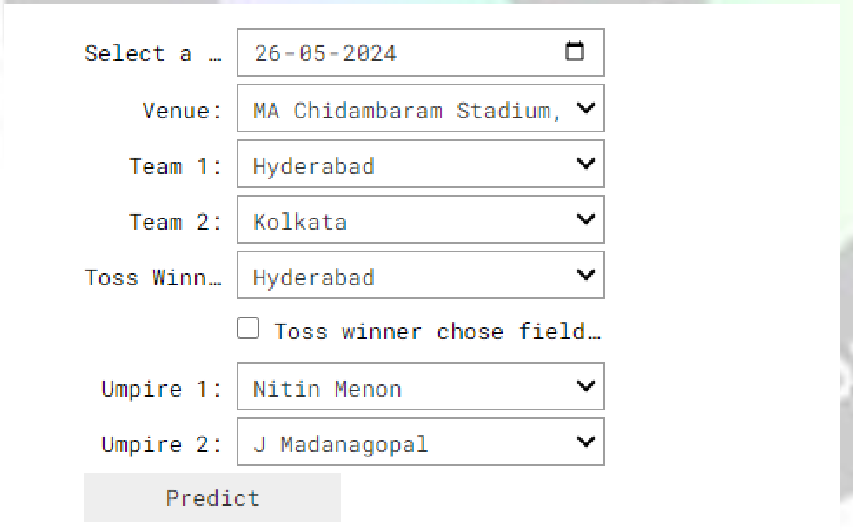
it predicts which team will going to win

RESULT



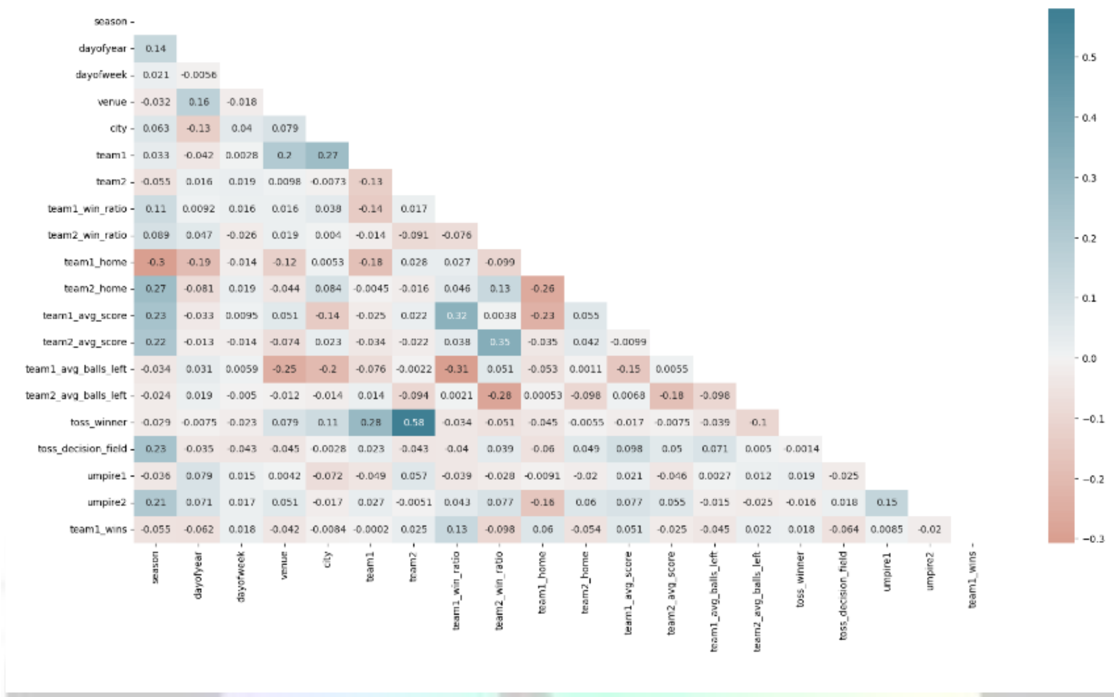
Select a …	dd-mm-yyyy
Venue:	M Chinnaswamy Stadium
Team 1:	Bangalore
Team 2:	Bangalore
Toss Winn…	Bangalore
	☐ Toss winner chose field…
Umpire 1:	Asad Rauf
Umpire 2:	Asad Rauf

AFTER ADDING DETAILS:



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

GRAPH



PICTURES

CONCLUSION

Utilizing the strength of data and cutting-edge algorithms, machine learning has completely changed how IPL predictions are made. Accurate forecasts of game results, player performances, and even tournament winners may be generated by analyzing past data, choosing pertinent attributes, and using a variety of machine-learning algorithms. Due to the inherent uncertainties in sports, no prediction model can guarantee 100% accuracy, but machine learning offers a data-driven approach that improves decision-making and gives the IPL another level of excitement.

Predicting a winner in a sport such as cricket is especially challenging and involves very complex processes. But with the introduction of machine learning, this can be made much easier and simpler. In this paper, various factors have been identified that contribute to the results of the Indian Premier League matches. Factors that have a major impact on the outcome of an IPL match include the teams playing, the venue, the city, the toss winner and the toss decision. We have analyzed IPL data sets and predicted game results based on player performance. The methods used in the work to obtain the final test are Logistic regression, Support Vector Machine (SVM), Decision tree, Random Forest classifier and K-nearest neighborhood. Random Forest classification (RFC) outperforms the other algorithm. As for the scope of the future, the focus can be on each

The important information regarding IPL score prediction and winning prediction system, that which parameters are required also the classifiers and algorithms. it helps in mathematical operation. Using all the information we have developed a website. for that the important work we have to do for the model is comparative analysis of machine learning techniques that is for score prediction the regressions and for winning prediction the analysis of classifiers. In Score Prediction analysis accuracy of Linear Regression is more than Ridge and Lasso Regression and in winning prediction analysis among SVC, Decision tree classifier and Random forest classifier, we got Random forest classifier accuracy more than other 2, with all 90%, 80%, 75%, 70% training data

ACKNOWLEDGEMENT

My sincere thanks to **IIDT BLACKBUCKS** for giving me the valuable knowledge on ARTIFICIAL INTELLIGENCE ,MACHINE LEARNING AND DATA SCIENCE[AIML-DS]

REFERENCES

[1] T. Singh, V. Singla and P. Bhatia, "Score and winning prediction in cricket through data mining," 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI), Faridabad, India, 2015, pp. 60-66, doi: 10.1109/ICSCTI.2015.7489605.

[2] J. Kumar, R. Kumar and P. Kumar, "Outcome Prediction of ODI Cricket Matches using Decision Trees and MLP Networks," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, 2018, pp. 343-347, doi: 10.1109/ICSCCC.2018.8703301.

[3] A. Kaluarachchi and S. V. Aparna, "CricAI: A classification based tool to predict the outcome in ODI cricket," 2010 Fifth International Conference on Information and Automation for Sustainability, Colombo, Sri Lanka, 2010, pp. 250-255, doi: 10.1109/ICIAFS.2010.5715668.

[4] A. I. Anik, S. Yeaser, A. G. M. I. Hossain and A. Chakrabarty, "Player's Performance Prediction in ODI Cricket Using Machine Learning Algorithms," 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018, pp. 500-505, doi: 10.1109/CEEICT.2018.8628118.

[5] N. Rodrigues, N. Sequeira, S. Rodrigues and V. Shrivastava, "Cricket Squad Analysis Using Multiple Random Forest Regression," 2019 1st International Conference on Advances in Information Technology (ICAIT), Chikmagalur, India, 2019, pp. 104-108, doi: 10.1109/ICAIT47043.2019.8987367.

[6] M. Jhawar and V. Pudi, "Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach", European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Riva del Garda, 2016

[7] A. I. Anik, S. Yeaser, A. G. M. I. Hossain and A. Chakrabarty, "Player's Performance Prediction in ODI Cricket Using Machine Learning Algorithms," 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018, pp. 500-505, doi: 10.1109/CEEICT.2018.8628118.

[8] Rameshwari Lokhande, P. M. Chawan "Live Cricket Score and Winning Prediction" Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Volume-5 | Issue1 , February 2018, URL: http://www.ijtrd.com/papers/IJTRD12180.pdf

[9] H. Barot, A. Kothari, P. Bide, B. Ahir and R. Kankaria, "Analysis and Prediction for the Indian Premier League," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-7, doi: 10.1109/INCET49848.2020.9153972.

[10] A. Basit, M. B. Alvi, F. H. Jaskani, M. Alvi, K. H. Memon and R. A. Shah, "ICC T20 Cricket World Cup 2020 Winner Prediction Using Machine Learning Techniques," 2020 IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan, 2020, pp. 1-6, doi: 10.1109/INMIC50486.2020.9318077.

[11] A. Bandulasiri, "Predicting the Winner in One Day International Cricket", Journal of Mathematical Sciences & Mathematics Education, Vol. 3, No. 1.

[12] Analysis and Prediction of Cricket Statistics using Data Mining Techniques Anurag Gangal VESIT, Mumbai Abhishek Talnikar VESIT, Mumbai Aneesh Dalvi VESIT, Mumbai Vidya Zope VESIT, Mumbai Aadesh Kulkarni VESIT, Mumbai.