IPL Score Prediction Using Machine Learning

Project Based Learning (PBL) Report
for the course
Machine Learning- 20CS41002

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By
Potharla Baby Shiva Naga Sriya(21R11A0595)
Karra Dinesh(2215A0510)
P Nathan Vishwas(21R11A0594)
Kandikatla Rakhi(21R11A0577)

Under the guidance of
E. Mahender
Associate Professor

Department of Computer Science and Engineering
Accredited by NBA

Geethanjali College of Engineering and Technology
(UGC Autonomous)
(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)
Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

December -2024

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. A. Sree Lakshmi, Professor**, Head of Department of Computer Science, Geethanjali College of Engineering and Technology, Cheeryal, whose motivation in the field of software development has made us to overcome all hardships during study and successful completion of project.

We would like to express our profound sense of gratitude to all for having helped us in completing this dissertation. We would like to express our deep-felt gratitude and sincere thanks to our guide **Mr. E.Mahender**, **Associate Professor**, Department of Computer Science, Geethanjali College of Engineering and Technology, Cheeryal, for his skillful guidance, timely suggestions and encouragement in completing this project.

We would like to express our sincere gratitude to our Principal **Prof. Dr. S. Udaya Kumar** for providing the necessary infrastructure to complete our project. We are also thankful to our Secretary **Mr. G.R. Ravinder Reddy** for providing an interdisciplinary & progressive environment.

Finally, we would like to express our heartfelt thanks to our parents who were very supportive both financially and mentally and for their encouragement to achieve our set goals.

Potharla Baby Shiva Naga Sriya (21R11A0595)

Karra Dinesh(22R15A0510)

P Nathan Vishwas(21R11A0594)

Kandikatla Rakhi(21R11A0577)

TABLE OF CONTENTS

1.**Introduction**

The Indian Premier League (IPL), one of the most exciting and popular T20 cricket leagues globally, has drawn significant attention from analysts and sports enthusiasts who aim to predict match outcomes and player performances. Machine learning (ML) provides a powerful tool to make these predictions more precise by learning patterns from historical data, which includes player stats, team dynamics, and external factors like weather and pitch conditions. This helps improve the viewing experience, supports betting strategies, and enhances fantasy sports analysis.

Creating an IPL score prediction model involves several steps: collecting and preprocessing historical data, identifying impactful features, training the model using ML algorithms (such as regression, decision trees, or gradient boosting), and evaluating its accuracy using performance metrics. Techniques like cross-validation and hyperparameter tuning help refine the model's predictive capabilities. By incorporating real-time data like current player form and weather conditions, these models can provide updated score forecasts for upcoming matches.

However, predicting scores in cricket comes with challenges due to the sport's inherent unpredictability, influenced by factors like sudden injuries, changing playing conditions, and strategic shifts. Ensuring that a model generalizes well to new data (avoiding overfitting) and maintaining high data quality are crucial for making reliable predictions. When done effectively, ML-based score prediction models can provide valuable insights and enhance sports analytics, offering fans and analysts a data-driven way to understand match outcomes.

About the project

The project aimed to develop a machine learning model to predict the total score of IPL (Indian Premier League) cricket matches using various match-related features. The process began with collecting historical IPL data, performing exploratory data analysis (EDA) to understand the distribution and relationships among features, and preprocessing the data through feature scaling and encoding. Multiple machine learning models, including Linear Regression, Random Forest Regressor, Support Vector Regressor (SVR), K-Nearest Neighbors Regressor (KNN), and a Neural Network, were trained and evaluated based on performance metrics like MAE, MSE, RMSE, and R-squared.

The neural network model, optimized with hyperparameter tuning via `RandomizedSearchCV`, emerged as the top performer due to its ability to learn complex data patterns effectively. To make the predictions more accessible, an interactive tool using `ipywidgets` was developed, allowing users to input match details and receive real-time score predictions from the trained model. This project demonstrated the value of applying machine learning to sports analytics, providing an innovative way to predict IPL match outcomes and enhancing the fan experience with data-driven insights.

Project outcomes and objectives

Project outcomes

1. Model Development and Evaluation: Various machine learning models were successfully trained and tested, including Linear Regression, Random Forest Regressor, Support Vector Regressor (SVR), K-Nearest Neighbors Regressor (KNN), and a Neural Network. The neural network model, after hyperparameter tuning using RandomizedSearchCV, emerged as the best performer, providing the most accurate predictions with strong evaluation metrics (e.g., low MAE, MSE, and RMSE).

2. Interactive Prediction Tool: An interactive tool was developed using ipywidgets, allowing users to input match-specific details (such as venue, batting team, bowling team, and players) and receive real-time score predictions from the best-trained neural network model. This feature enhanced the practical utility of the project, making it accessible for real-time use by fans and analysts.

3. Insights into Data and Performance: The project provided valuable insights into how different features (such as venue, player form, and team composition) influence match scores. The exploration phase revealed patterns in historical

match data, contributing to a deeper understanding of the factors affecting cricket scores.

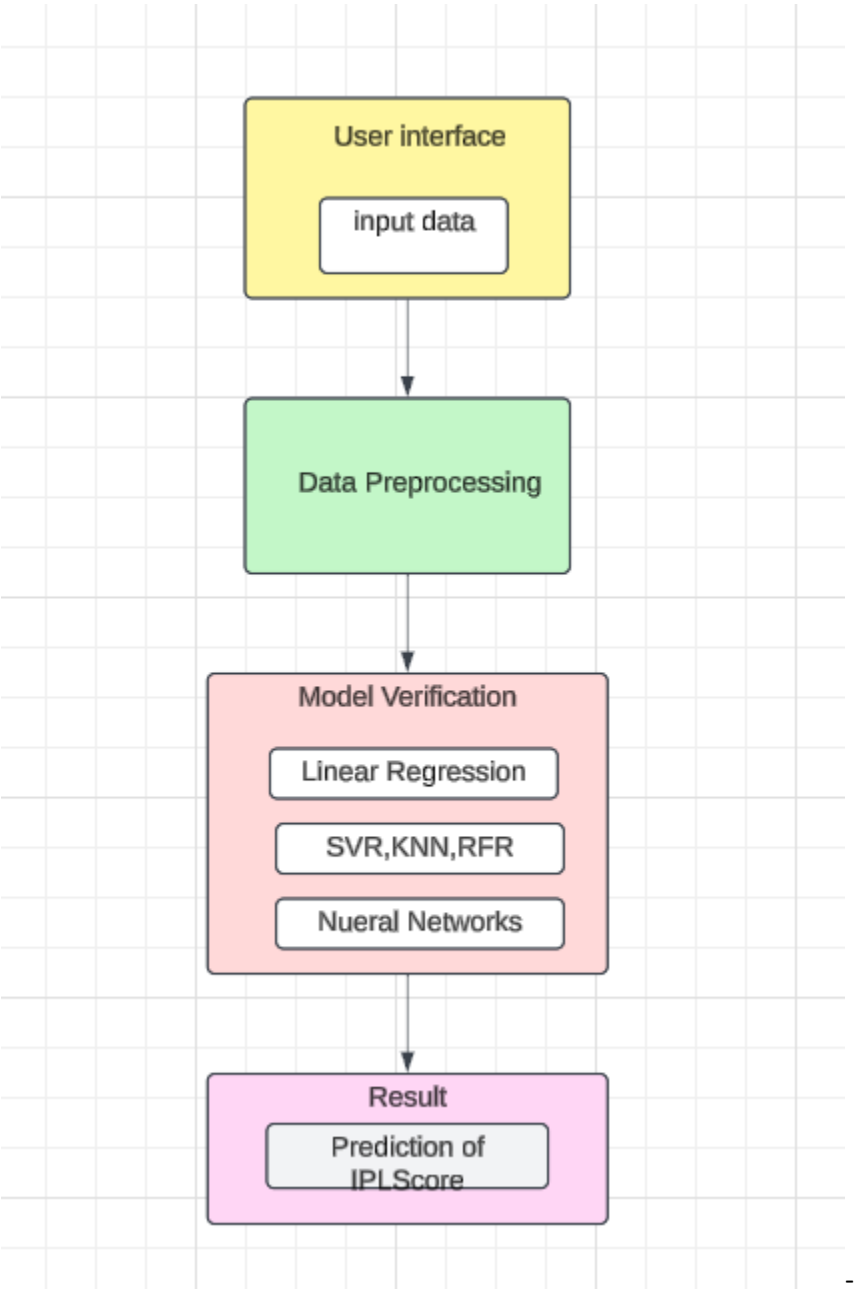
4. Application to Sports Analytics: The project demonstrated the potential of machine learning in sports analytics, highlighting its ability to deliver data-driven predictions that can enhance the fan experience, support betting strategies, and inform team strategies. It showcased the practical application of ML in analyzing cricket matches and provided a foundation for further enhancements or expansion to include more complex predictions.

Project Objectives

1. Develop Accurate Score Predictions: Create a machine learning model to reliably predict IPL match scores using historical and real-time data.
2. Support Cricket Analytics: Assist fans, analysts, and fantasy league players in making data-driven decisions through score predictions and insights.
3. Leverage Historical Data: Analyze past IPL matches to uncover patterns and trends that influence match outcomes.
4. Incorporate Real-Time Inputs: Enable dynamic predictions based on live match details like team line-ups, pitch reports, and weather conditions.
5. Build an Interactive Tool: Design a user-friendly interface to input match details and display predictions effectively.

2. System Design

- System Architecture



1. User Interface Layer

- **Input Data**: Users provide input data through a user-friendly interface, which includes match details such as venue, batting team, bowling team, batsman, bowler, and other real-time match-specific information.
- This layer facilitates interaction between the user and the backend system, ensuring ease of use.

2. Data Preprocessing Layer

- **Purpose**: Prepares the raw data for analysis and ensures compatibility with machine learning models.
- **Key Steps**:
    - **Data Cleaning**: Handles missing values, removes irrelevant data, and resolves inconsistencies.
    - **Feature Scaling**: Standardizes numerical data using techniques like MinMaxScaler to ensure uniform input ranges for models.
    - **Encoding**: Converts categorical data (like team names or venue) into numerical formats using methods like LabelEncoder or one-hot encoding.
    - **Feature Engineering**: Adds meaningful features such as recent player performance, pitch statistics, or weather impact to improve model accuracy.

3. Model Verification and Training Layer

- **Objective**: Compares different machine learning models to determine the best-performing algorithm for IPL score prediction.
- **Models Evaluated**:
    - **Linear Regression**: A basic model to set a baseline for comparison.
    - **Support Vector Regressor (SVR)**: Handles non-linear relationships in data.
    - **K-Nearest Neighbors (KNN)**: Predicts based on similarity with nearest data points.
    - **Random Forest Regressor (RFR)**: Uses ensemble learning to handle feature interactions and reduce overfitting.

    - **Neural Networks**: A deep learning model that provides high accuracy after hyperparameter tuning.

- **Training Environment**: Uses libraries like TensorFlow, Keras, and scikit-learn for model development and training.
- **Evaluation**: Metrics like MAE, MSE, RMSE, and R-squared are used to evaluate and compare the models.
- **Best Model**: The optimized neural network is selected for its superior accuracy and ability to handle complex relationships in data.

4. Prediction and Results Layer

- **Prediction**:
    - Once the best model (e.g., Neural Network) is trained, it is used to predict IPL match scores based on new input data.
    - Real-time match data is processed and passed to the model, which outputs the predicted score.
- **Result Display**:
    - The predictions are displayed to the user in a meaningful way, including visualizations like graphs or charts if needed.

5. User Interaction Layer

- **Interactive Dashboard**:
    - Provides an intuitive and visually appealing interface for users to input match details and view predictions.
    - Built using tools like ipywidgets or a web framework such as Flask for seamless interaction.
- **Output Display**:
    - The final predicted score is presented, along with additional insights, such as the impact of specific players or teams on the prediction.

-Modules

1. Input Module (Data Input and User Interaction):

- **Purpose**: This module allows users to provide input related to IPL match details, such as venue, batting team, bowling team, and player-specific information, which are essential for score prediction.
- **Process**: Users can enter match details through an interactive web interface or a notebook-based dashboard using libraries like ipywidgets or Streamlit. This data is then collected and formatted for further processing.
- **Responsibilities**: Collect user input, validate the format, and prepare data for preprocessing.

2. Data Preprocessing Module:

- **Purpose**: The preprocessing module prepares the input data for the model by ensuring consistency and compatibility with the model's input requirements.
- **Process**: The input data is transformed and scaled using preprocessing techniques such as LabelEncoder for categorical features and MinMaxScaler for numerical features. Missing values, if any, are handled, and the data is organized into the required format.
- **Responsibilities**: Encode categorical data, scale numerical data, handle missing values, and format data for model training and prediction.

3. Model Selection and Training Module:

- **Purpose**: This module handles the training and evaluation of different machine learning models for score prediction, helping to identify the best-performing model.
- **Process**:
    - **Training**: Multiple models are trained using historical IPL match data, including Linear Regression, Random Forest Regressor, Support Vector Regressor (SVR), K-Nearest Neighbors Regressor (KNN), and a custom-built Neural Network.
    - **Evaluation**: Each model's performance is evaluated using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared score to assess prediction accuracy.

    - **Hyperparameter Tuning**: The best models undergo hyperparameter tuning using RandomizedSearchCV or GridSearchCV to optimize performance.

- **Responsibilities**: Train and fine-tune models, select the best-performing model based on evaluation metrics, and save the trained model for use in real-time predictions.

4. Result Display Module:

- **Purpose**: This module is responsible for presenting the predictions generated by the trained model in an intuitive manner for the end-users.
- **Process**:
    - **Prediction Generation**: The best model takes user inputs (preprocessed data) and predicts the score for the upcoming IPL match.
    - **Display**: The prediction results are formatted and displayed on a web interface or notebook interface. This can include visualizations like score charts or tables for a more comprehensive view.
    - **User Interaction**: Feedback and additional context may be provided to assist users in understanding the prediction and making data-driven decisions.
- **Responsibilities**: Display the predicted scores, present relevant statistics, and ensure the output is user-friendly and informative.

- **Backend Design Code**

```
from flask import Flask, render_template, request, jsonify
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import tensorflow as tf
from tensorflow import keras
import os
import logging
logging.basicConfig(level=logging.DEBUG)
app = Flask(__name__)
model_path = os.path.join(os.getcwd(), 'model.h5')
data_path = os.path.join(os.getcwd(), 'ipl_data.csv')
try:
    ipl = pd.read_csv(data_path)
except FileNotFoundError:
    logging.error(f"Data file not found at {data_path}")
df = ipl.drop(['date', 'runs', 'wickets', 'overs', 'runs_last_5',
'wickets_last_5', 'mid', 'striker', 'non-striker'], axis=1)
X = df.drop(['total'], axis=1)
y = df['total']
venue_encoder = LabelEncoder()
```

```
batting_team_encoder = LabelEncoder()
bowling_team_encoder = LabelEncoder()
striker_encoder = LabelEncoder()
bowler_encoder = LabelEncoder()

X['venue'] = venue_encoder.fit_transform(X['venue'])
X['bat_team'] = batting_team_encoder.fit_transform(X['bat_team'])
X['bowl_team'] =
bowling_team_encoder.fit_transform(X['bowl_team'])
X['batsman'] = striker_encoder.fit_transform(X['batsman'])
X['bowler'] = bowler_encoder.fit_transform(X['bowler'])
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
try:
    model = keras.models.load_model(model_path)
except Exception as e:
    logging.error(f"Error loading the model: {e}")
    raise
@app.route('/')
def index():
    try:
```

```
        return render_template('index.html',
venues=list(ipl['venue'].unique()),
                               batting_teams=list(ipl['bat_team'].unique()),
                               bowling_teams=list(ipl['bowl_team'].unique()),
                               batsmen=list(ipl['batsman'].unique()),
                               bowlers=list(ipl['bowler'].unique()))
    except Exception as e:
        logging.error(f"Error rendering the index page: {e}")
        return "An error occurred while loading the page."
@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.json
        input_data = [
            venue_encoder.transform([data['venue']])[0],
            batting_team_encoder.transform([data['batting_team']])[0],

bowling_team_encoder.transform([data['bowling_team']])[0],
            striker_encoder.transform([data['striker']])[0],
            bowler_encoder.transform([data['bowler']])[0]
        ]
        input_data = scaler.transform([input_data])
```

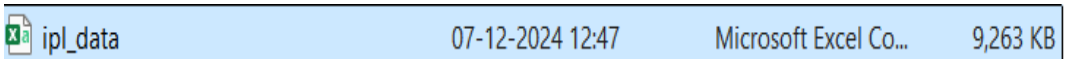
```
        prediction = model.predict(input_data)
        predicted_score = int(prediction[0, 0])
        return jsonify({'predicted_score': predicted_score})
    except Exception as e:
        logging.error(f"Error during prediction: {e}")
        return jsonify({'error': 'Error predicting score. Please check your input data.'})

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

3. Implementation

The project is implemented using Python and involves the following steps:

1. Data Collection and Preprocessing:

   Data set Used:

   The IPL dataset contains **76,014 rows** and **15 columns**. Here's a brief overview:

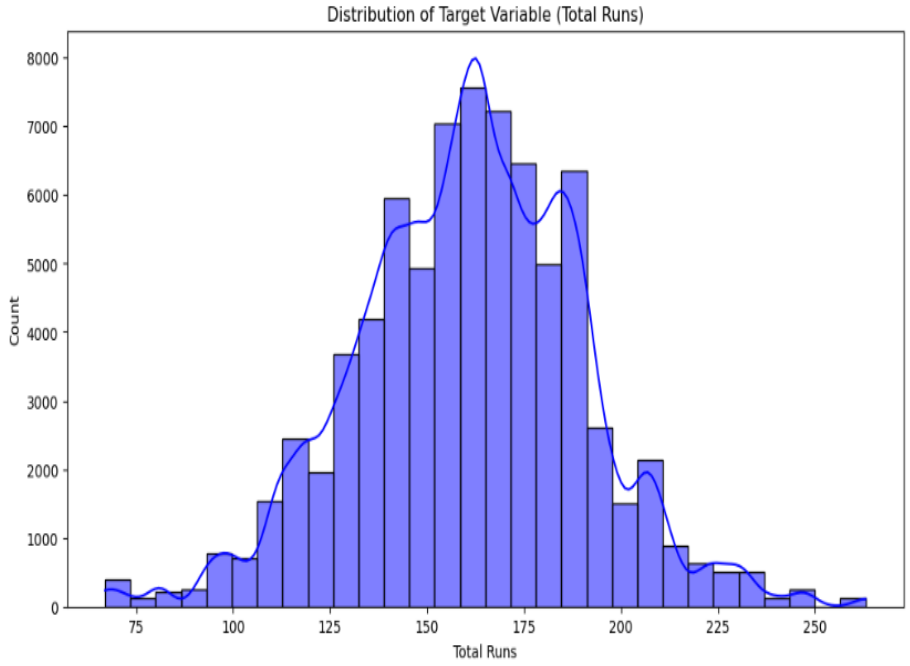
   - **mid**: Match ID (unique identifier for matches).
   - **date**: Date of the match.
   - **venue**: Stadium where the match was played.
   - **bat_team**: Batting team name.
   - **bowl_team**: Bowling team name.
   - **batsman**: Name of the batsman on strike.
   - **bowler**: Name of the bowler delivering the ball.
   - **runs**: Runs scored on the delivery.
   - **wickets**: Total wickets fallen at the given point in the match.
   - **overs**: Overs completed in the match.
   - **runs_last_5**: Runs scored in the last 5 overs.
   - **wickets_last_5**: Wickets lost in the last 5 overs.
   - **striker**: Current runs scored by the batsman on strike.
   - **non-striker**: Runs scored by the batsman at the non-striker's end.
   - **total**: Total runs scored by the batting team in the match.

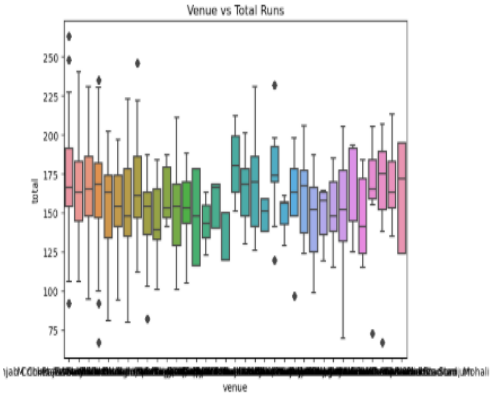
   Preprocess the data to handle missing values, encode categorical variables, and scale numerical data using libraries like pandas, numpy, and sklearn.

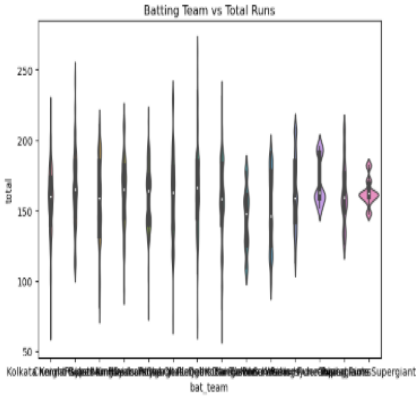
2. Exploratory Data Analysis (EDA):

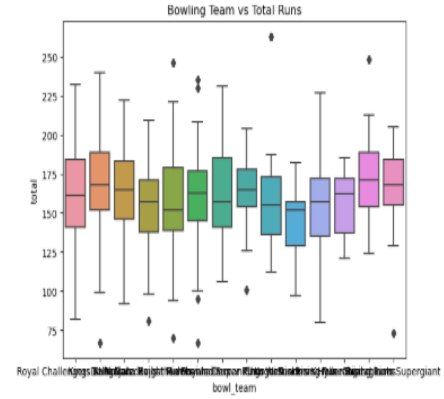
- Analyze data distribution, correlations, and feature importance using visualization tools like matplotlib and seaborn.

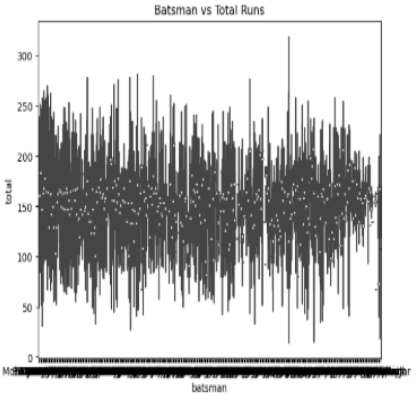
```
Text(0, 0.5, 'Count')
```







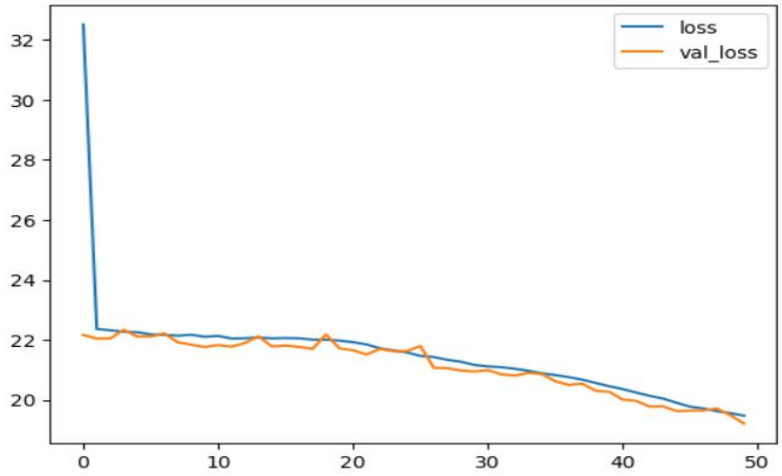




3. Model Selection and Training:

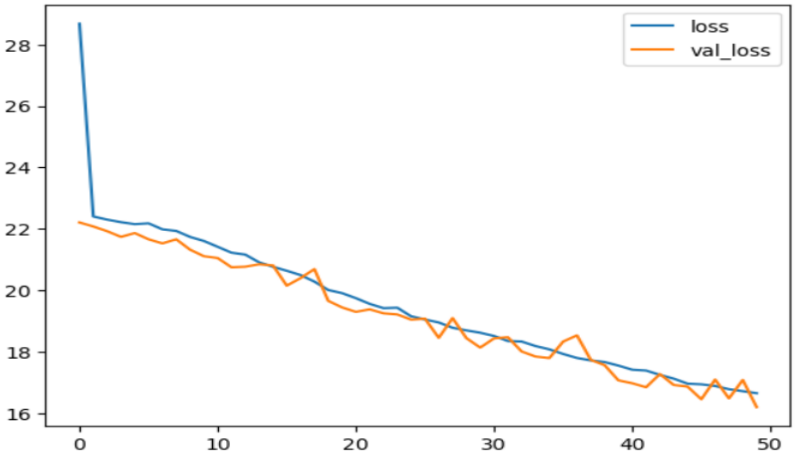
   - Train multiple machine learning models (Linear Regression, Random Forest, SVR, KNN, Neural Networks) using libraries like scikit-learn and tensorflow.

     Linear Regression




```
Mean Absolute Error (MAE): 19.708150461937507
Mean Squared Error (MSE): 717.521534733854
R-squared (R2) Score: 0.14923469355266783
```

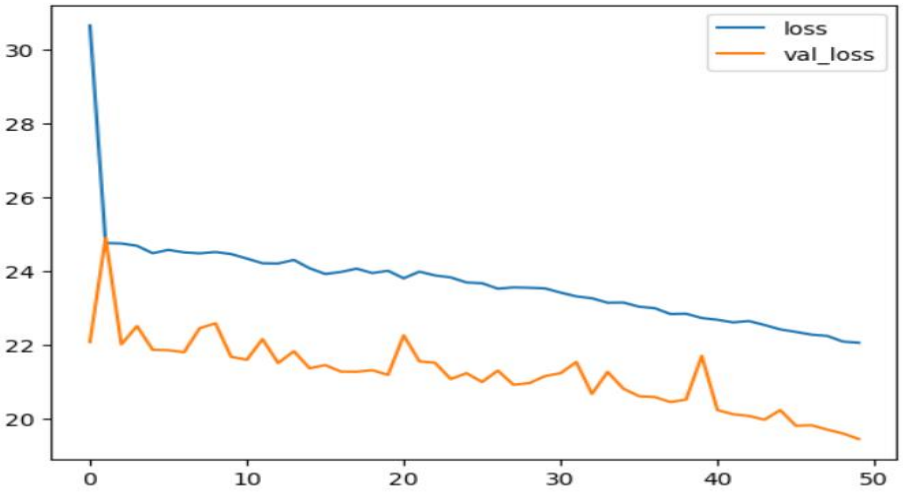
     Random Forest




```
Mean Absolute Error (MAE): 16.687844785737564
Mean Squared Error (MSE): 564.6388625738492
R-squared (R2) Score: 0.3305076827723412
```

SVR

```
Mean Absolute Error (MAE): 19.960248648809305
Mean Squared Error (MSE): 703.4661732258477
R-squared (R2) Score: 0.16590013613763832
```

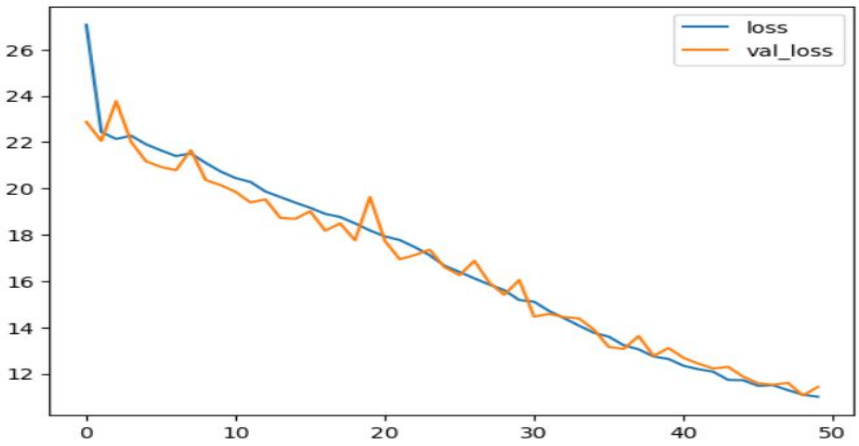


KNN


```
713/713 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step
Mean Absolute Error (MAE): 20.563513565983488
Mean Squared Error (MSE): 737.3897240990281
R-squared (R2) Score: 0.12567698079344625
```

Nueral Networks

```
Mean Absolute Error (MAE): 11.906202845290103
Mean Squared Error (MSE): 352.1048339262671
R-squared (R2) Score: 0.5825092872676203
```



- The best-performing model was a neural network trained with the optimal hyperparameters found via RandomizedSearchCV. It was evaluated for performance metrics such as MAE, MSE, RMSE, and R2 score.

4. Model Optimization:
    - Optimize hyperparameters using techniques like grid search or randomized search to improve model accuracy.
5. Integration with User Interface:
    - Develop a user-friendly interface using ipywidgets or Flask to input match data and display predictions.

- Modules implementation

User Interface Module: The web application is built using Flask, a lightweight Python web framework. After opening the link the venue,bowler, batsman, team names of 2 IPLl teams are taken as input.

```
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Running Flask application on localhost

Data Preprocessing Module: Input data is preprocessed

Prediction Module: Input data is passed to the Neural networks model predicts the IPL Score.

Result Module: the result module displays the result of the predicted IPL score .

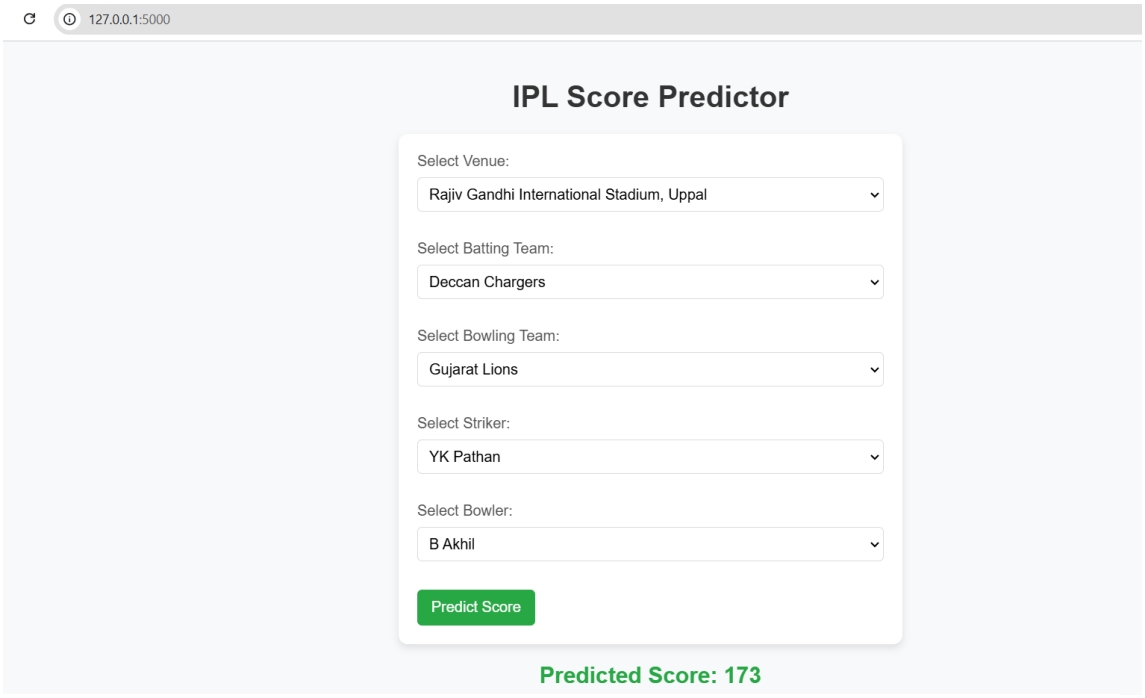
- Sample code

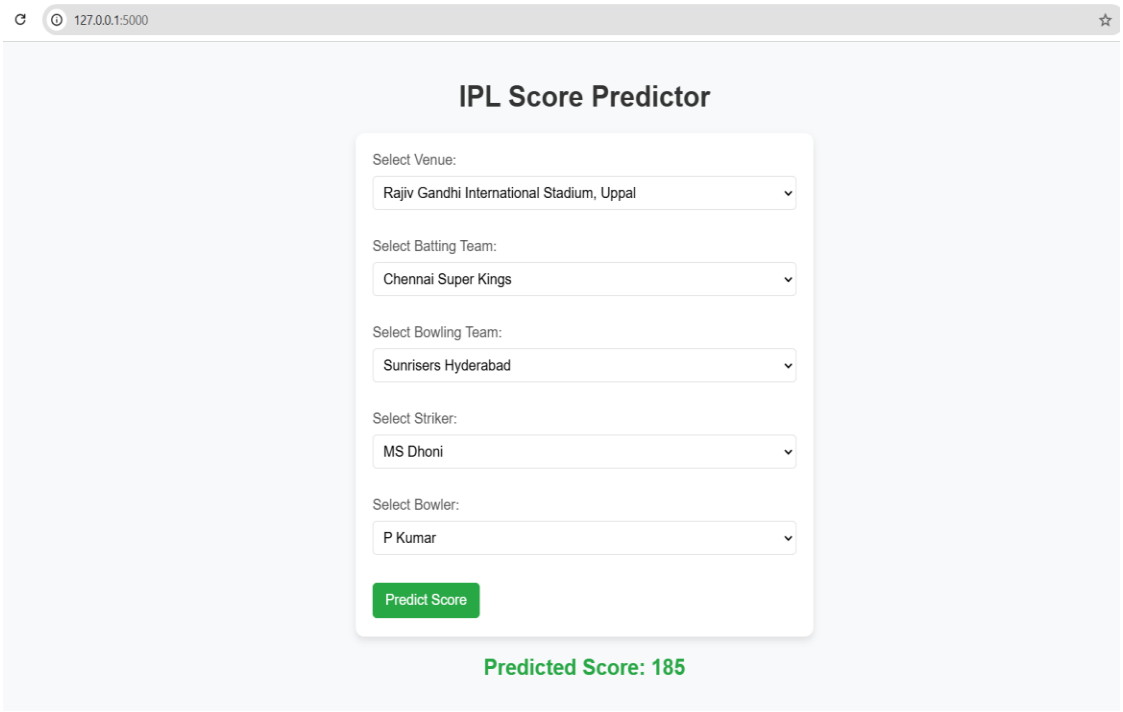
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
import keras
import tensorflow as tf
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

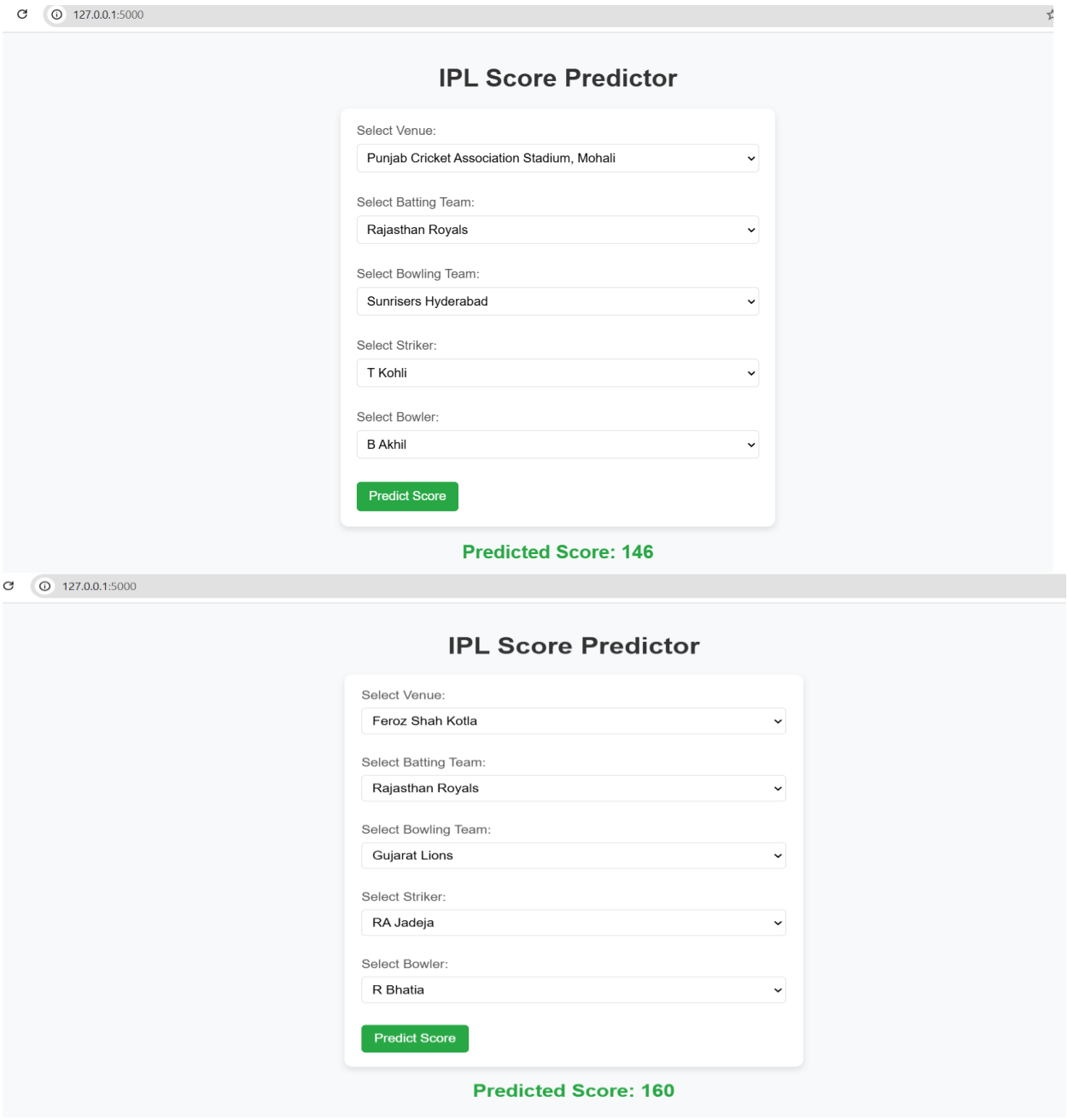
```
import ipywidgets as widgets

from IPython.display import display, clear_output

import warnings

df = ipl.drop(['date', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5', 'mid', 'striker', 'non-striker'], axis=1)

numeric_columns = df.select_dtypes(include=[np.number]).columns.tolist()

X = df.drop(['total'], axis=1)

y = df['total']

plt.figure(figsize=(12, 8))

plt.figure(figsize=(12, 6))

sns.histplot(y, bins=30, kde=True, color='blue')

plt.title('Distribution of Target Variable (Total Runs)')

plt.xlabel('Total Runs')

plt.ylabel('Count')

plt.figure(figsize=(16, 10))

plt.subplot(2, 2, 1)

sns.boxplot(x='venue', y='total', data=df)

plt.title('Venue vs Total Runs')


plt.subplot(2, 2, 2)

sns.violinplot(x='bat_team', y='total', data=df)

plt.title('Batting Team vs Total Runs')


plt.subplot(2, 2, 3)

sns.boxplot(x='bowl_team', y='total', data=df)

plt.title('Bowling Team vs Total Runs')
```

```
plt.subplot(2, 2, 4)
sns.violinplot(x='batsman', y='total', data=df)
plt.title('Batsman vs Total Runs')

plt.tight_layout()
venue_encoder = LabelEncoder()
batting_team_encoder = LabelEncoder()
bowling_team_encoder = LabelEncoder()
striker_encoder = LabelEncoder()
bowler_encoder = LabelEncoder()X['venue'] = venue_encoder.fit_transform(X['venue'])
X['bat_team'] = batting_team_encoder.fit_transform(X['bat_team'])
X['bowl_team'] = bowling_team_encoder.fit_transform(X['bowl_team'])
X['batsman'] = striker_encoder.fit_transform(X['batsman'])
X['bowler'] = bowler_encoder.fit_transform(X['bowler'])
# Train-test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- Sample output screenshots /Results







The above output shows the IPL Scor predictions of the batsman basing on t the bowler,venue and team.

4.Conclusion

The IPL score prediction web application, built using Flask, provides a powerful and user-friendly tool for cricket enthusiasts, analysts, and stakeholders to gain insights into match outcomes. By leveraging a trained machine learning model, the application processes user inputs, preps the data, and delivers real-time predictions on the potential scores of batting teams. The integration of preprocessing techniques, such as feature encoding and scaling, ensures that the input data is in the optimal format for accurate predictions. Additionally, the application displays results in a visually engaging manner, with optional visualizations to support understanding and analysis. Error handling mechanisms enhance the user experience by providing clear feedback when issues occur. Deployed on cloud platforms, the application offers seamless accessibility on both desktop and mobile devices, making it a valuable resource for informed decision-making in cricket analytics. Overall, this project demonstrates the practical application of machine learning in sports analytics, contributing to better insights and predictions for IPL matches.

5. References

https://www.iplt20.com/

https://ieeexplore.ieee.org/document/10134747