# Title: CRICKET SCORE PREDICTION BY LINEAR REGRESSION

A

ADM Course Project Report

in partial fulfilment of the degree

**Bachelor of Technology**
in
**Computer Science & Engineering**


**By**

| | |
|---|---|
| **N.Prashanth** | **2303A51063** |
| **N.Akhil** | **2303A51065** |
| **D.Nithish** | **2303A51048** |
| **D.Vinay** | **2303A51050** |
| **K.Vishwesh reddy** | **2303A51057** |

Under the guidance of

**Bediga Sharan**
**Assistant Professor**


**Submitted to**

**School of Computer Science and Artificial Intelligence**

SR UNIVERSITY

\

# DEPARTMENT OFCOMPUTERSCIENCE& ENGINEERING

## CERTIFICATE

This is to certify that the **Application of Data Mining – Course Project** Report entitled **"CRICKET SCORE PREDICTION"** is a record of bonafide work carried out by the student(s) **"Prashanth,Akhil,Nithish,Vinay,vishweshreddy",**bearing,Hallticket,No(s)**2303A51063,1065,1048, 1050,1057during** the academic year 2024-25 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University, Warangal.

**Supervisor**
(Mr. Bediga Sharan)
Assistant Professor

**Head of the Department**
(Dr. M. Sheshikala)
Professor

+

# Overview of the Project ：

**Objective:**

The goal of this project is to predict the final score of a cricket innings (usually in T20 or ODI formats) based on available match data at a certain point (e.g., after 6 or 10 overs). By analyzing historical cricket match data, we aim to build a model that can forecast the total runs a team might score in their innings.

---

📊 Key Components:

1. **Data Collection**
   - Datasets of past cricket matches (commonly from Kaggle or cricsheet.org)
   - Key features include:
     - Batting team, bowling team
     - Venue
     - Current runs, wickets, overs
     - Run rate, partnership info, recent overs stats
2. **Data Preprocessing**
   - Cleaning missing or inconsistent values
   - Encoding categorical variables (like team names, venue)
   - Feature engineering (e.g., balls left, wickets in hand, etc.)
3. **Model Building**
   - Regression models used to predict numerical scores:
     - Linear Regression
     - Decision Tree Regressor
     - Random Forest
     - XGBoost
   - Evaluation using metrics like RMSE, MAE, and $R^2$ score
4. **Model Training and Testing**
   - Data split into training and test sets
   - Cross-validation to avoid overfitting
   - Hyperparameter tuning (e.g., using GridSearchCV)
5. **Prediction & Visualization**
   - Input: Current match scenario (overs completed, runs, wickets)
   - Output: Predicted final score
   - Visual tools like matplotlib/seaborn to compare actual vs. predicted scores

# TABLE OF CONTENTS



---

## 1.OBJECTIVE OF THE PROJECT:

The primary objective of the Score Prediction project is to develop a model that can accurately predict scores for a given event, match, or competition based on historical data and various influencing factors. The project aims to leverage machine learning techniques and statistical analysis to generate reliable predictions. Key objectives include:

**Developing a Predictive Model:** To build a model that can predict scores in real-time, based on input features such as past performance, team statistics, player data, and other relevant variables.

**Analyzing Historical Data:** To analyze historical data of games or competitions to identify patterns and trends that influence the outcome of a match or event, and use this data to train the model.

**Improving Prediction Accuracy:** To evaluate different machine learning algorithms and techniques to find the most accurate and reliable method for score prediction.

**Providing Insights:** To offer actionable insights, such as key factors affecting the outcome, which could help teams, coaches, or analysts make informed decisions.

**Testing and Evaluation:** To test the model on unseen data, evaluate its performance, and make necessary adjustments to improve the prediction accuracy.

## 3.DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT :

**Score Prediction:** Forecasting the final score of an event (e.g., a sports match) using historical data and features that affect the outcome.

**Historical Data:** Past match results and relevant statistics (teams, players, weather, etc.) used to train the prediction model.

**Features (Variables):** Input variables like team stats, player performance, and environmental factors that help predict the score.

**Target Variable (Score):** The outcome the model aims to predict, such as the final score of the match.

**Machine Learning Model:** An algorithm (e.g., Linear Regression, Random Forest) that uses historical data to make predictions.

**Training Data:** The portion of data used to train the model and teach it patterns and relationships.

**Testing Data:** Data used to evaluate how well the trained model predicts outcomes on unseen data.

**Evaluation Metrics:** Measures like MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) to assess the model's performance.

**Overfitting:** When the model performs well on training data but poorly on testing data due to memorizing rather than generalizing.

**Underfitting:** When the model is too simple to capture the patterns in the data, leading to poor performance on both training and testing data.

**Cross-Validation:** A method to assess the model's performance by splitting the data into multiple subsets for training and testing.

**Hyperparameter Tuning:** The process of adjusting model parameters (e.g., learning rate) to improve performance.

**Prediction Interval:** A range within which the predicted score is expected to fall, providing uncertainty estimation.

## 3.IMPLEMENTATION:

The Score Prediction system was built using the following steps:

### 3.1 Tools and Technologies

**Programming Language:** Python

**Libraries:** Pandas (data manipulation), Scikit-learn (modeling), Matplotlib/Seaborn (visualization)

**Data Sources:** Historical match data (from APIs or web scraping)

### 3.2 Data Collection and Preprocessing

**Data Collection**: Gathered historical match and player data.

**Data Preprocessing:** Handled missing data, encoded categorical variables, and scaled numerical features. New features like team form and head-to-head stats were created.

### 3.3 Model Development and Training

**Model Selection:** Tested Linear Regression, Random Forest, and Gradient Boosting for score prediction.

**Training:** Split data into training and testing sets (80-20 split), trained models, and optimized hyperparameters.

### 3.4 Model Evaluation

**Evaluation Metrics**: Used MAE, RMSE, and $R^2$ to assess model performance.

**Cross-Validation:** Employed K-fold cross-validation to ensure the model generalizes well.

### 3.5 Prediction and Results

**Real-time Prediction**: The model predicts scores based on user input (team, match details).

**Deployment:** Deployed the model via a web application or API for real-time use.

### 3.6 Challenges and Solutions

**Data Quality:** Managed missing data with imputation.

**Overfitting:** Mitigated by using cross-validation and regularization.

**Model Complexity:** Simpler models like Random Forest provided better results.

```
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
```
[1] ✓ 13.5s Python

```
import kagglehub
ronikdedhia_ipl_first_innings_score_path = kagglehub.dataset_download('ronikdedhia/ipl-first-innings-score')
print('Data source import complete.')
```
[2] ✓ 2.1s Python

```
Data source import complete.
```

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```
[3] ✓ 0.0s Python

```
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```
[4] ✓ 0.0s Python

```
import pandas as pd
df = pd.read_csv(r'C:\Users\akhil\OneDrive\Desktop\AKHIL\ADM-PROJECT\ipl_data.csv')
df.head(5)
```
[5] ✓ 0.1s Python

| | mid | date | venue | bat_team | bowl_team | batsman | bowler | runs | wickets | overs | runs_last_5 | wickets_last_5 | striker | non-striker | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | SC Ganguly | P Kumar | 1 | 0 | 0.1 | 1 | 0 | 0 | 0 | 222 |
| 1 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 1 | 0 | 0.2 | 1 | 0 | 0 | 0 | 222 |
| 2 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.2 | 2 | 0 | 0 | 0 | 222 |
| 3 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.3 | 2 | 0 | 0 | 0 | 222 |
| 4 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.4 | 2 | 0 | 0 | 0 | 222 |

**1. Importing and Understanding the Data:**

- The dataset is loaded using **Pandas**, providing a tabular view of past cricket match data.
- Initial exploration includes .head(), .info(), and .describe() to inspect column types, missing values, and basic statistics.
- Key features typically include: *batting team, bowling team, overs, runs, wickets, venue*, etc.

**2. Handling Missing Values:**

- Rows with missing critical values are dropped or imputed based on domain knowledge.
- Categorical features like team names or venues are cleaned to ensure consistency.

**3. Feature Encoding:**

- Categorical columns such as *batting team, bowling team*, and *venue* are transformed using **One-Hot Encoding** to ensure compatibility with ML models.

```
columns_to_remove = ['mid', 'venue', 'batsman', 'bowler', 'striker', 'non-striker']
print('Before removing unwanted columns: {}'.format(df.shape))
df.drop(labels=columns_to_remove, axis=1, inplace=True)
print('After removing unwanted columns: {}'.format(df.shape))
```

Python

```
Before removing unwanted columns: (76014, 15)
After removing unwanted columns: (76014, 9)
```

```
df['bat_team'].unique()
```

Python

```
array(['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
       'Mumbai Indians', 'Deccan Chargers', 'Kings XI Punjab',
       'Royal Challengers Bangalore', 'Delhi Daredevils',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Sunrisers Hyderabad',
       'Rising Pune Supergiants', 'Gujarat Lions',
       'Rising Pune Supergiant'], dtype=object)
```

```
consistent_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
                    'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',
                    'Delhi Daredevils', 'Sunrisers Hyderabad']
```

Python

```
print('Before removing inconsistent teams: {}'.format(df.shape))
df = df[(df['bat_team'].isin(consistent_teams)) & (df['bowl_team'].isin(consistent_teams))]
print('After removing inconsistent teams: {}'.format(df.shape))
```

Python

```
Before removing inconsistent teams: (76014, 9)
After removing inconsistent teams: (53811, 9)
```

```
df['bat_team'].unique()
```

Python

```
array(['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
       'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',
       'Delhi Daredevils', 'Sunrisers Hyderabad'], dtype=object)
```

```
print('Before removing first 5 overs data: {}'.format(df.shape))
df = df[df['overs']>=5.0]
print('After removing first 5 overs data: {}'.format(df.shape))
```

Python

```
Before removing first 5 overs data: (53811, 9)
After removing first 5 overs data: (40108, 9)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(100, 200))
df['scaled_runs'] = scaler.fit_transform(df[['runs']])
total_scaled_runs = df.groupby('bat_team')['scaled_runs'].sum().reset_index()
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=total_scaled_runs, x='bat_team', y='scaled_runs', ci=None, palette='viridis') # Using total_scaled_runs and adding
plt.title('Total Runs by Batting Team (Scaled)', fontsize=16)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.xlabel('Batting Team', fontsize=14)
plt.ylabel('Total Scaled Runs (100-200)', fontsize=14)
plt.tight_layout()
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='bottom',
                xytext=(0, 5), textcoords='offset points', fontsize=10, color='black')

plt.show()
```

**1. Runs Distribution Analysis:**

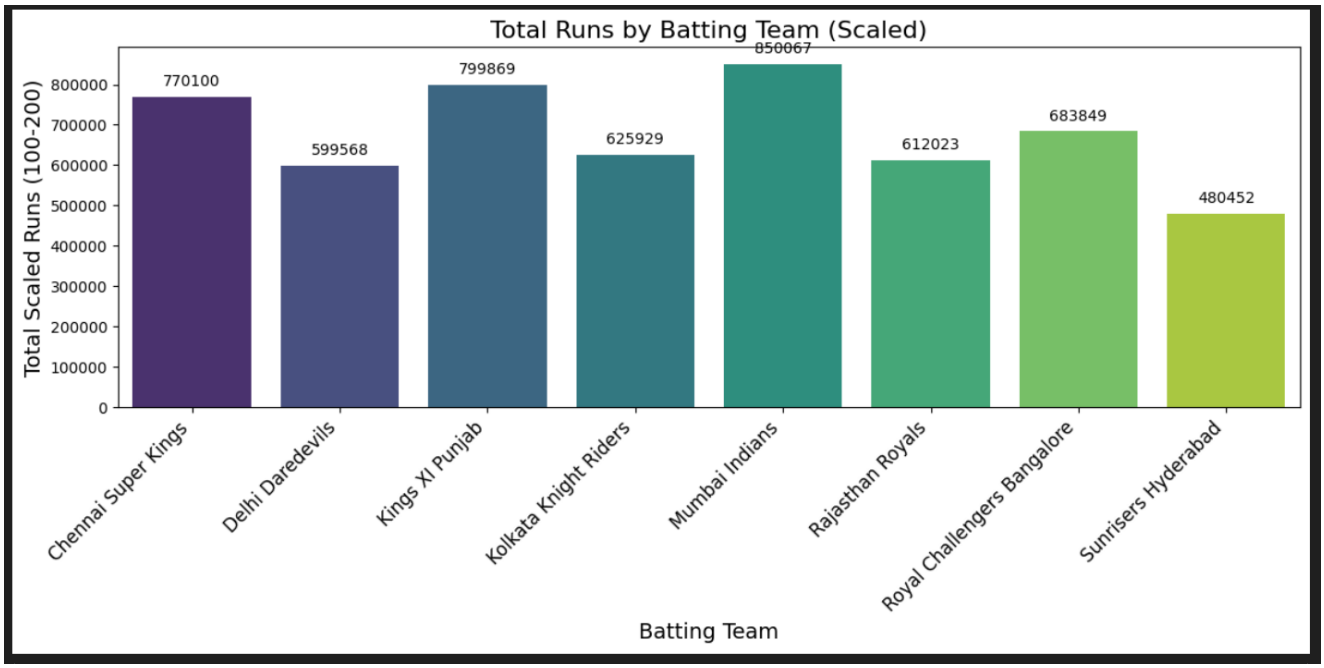
- A **histogram** is plotted using matplotlib to visualize how the total runs are distributed across the dataset.
- Helps in identifying any skewness or anomalies.

**2. Runs Over Time (Overs):**

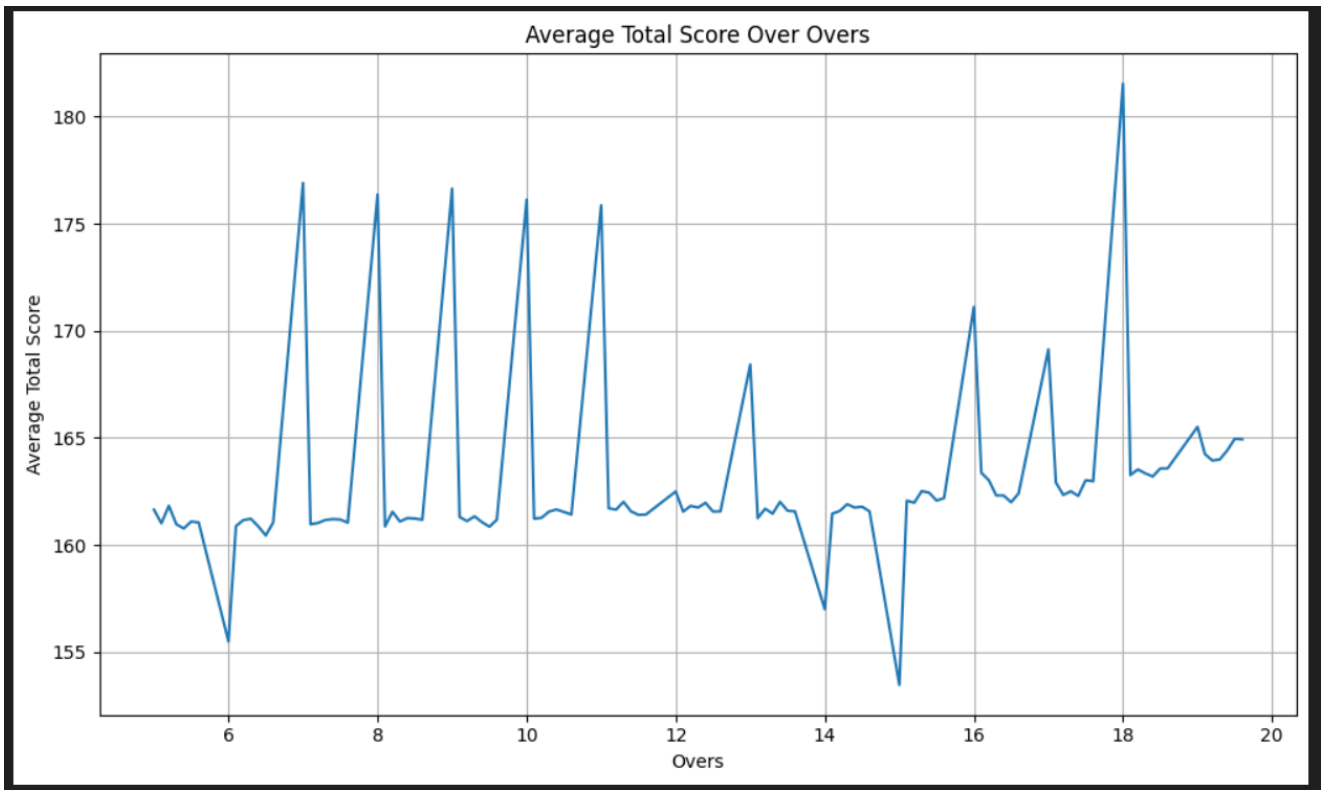
- A **line plot** shows the average runs scored over each over (1 to 20), indicating acceleration patterns across innings.

**3. Team-wise Performance:**

- **Box plots** compare runs scored by different batting teams to visualize team consistency and outliers.




```
plt.figure(figsize=(10,6))
sns.lineplot(data=df, x='overs', y='total', estimator='mean', ci=None)
plt.title('Average Total Score Over Overs')
plt.xlabel('Overs')
plt.ylabel('Average Total Score')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
    print("Before converting 'date' column from string to datetime object: {}".format(type(df.iloc[0,0])))
    df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
    print("After converting 'date' column from string to datetime object: {}".format(type(df.iloc[0,0])))
```

Python

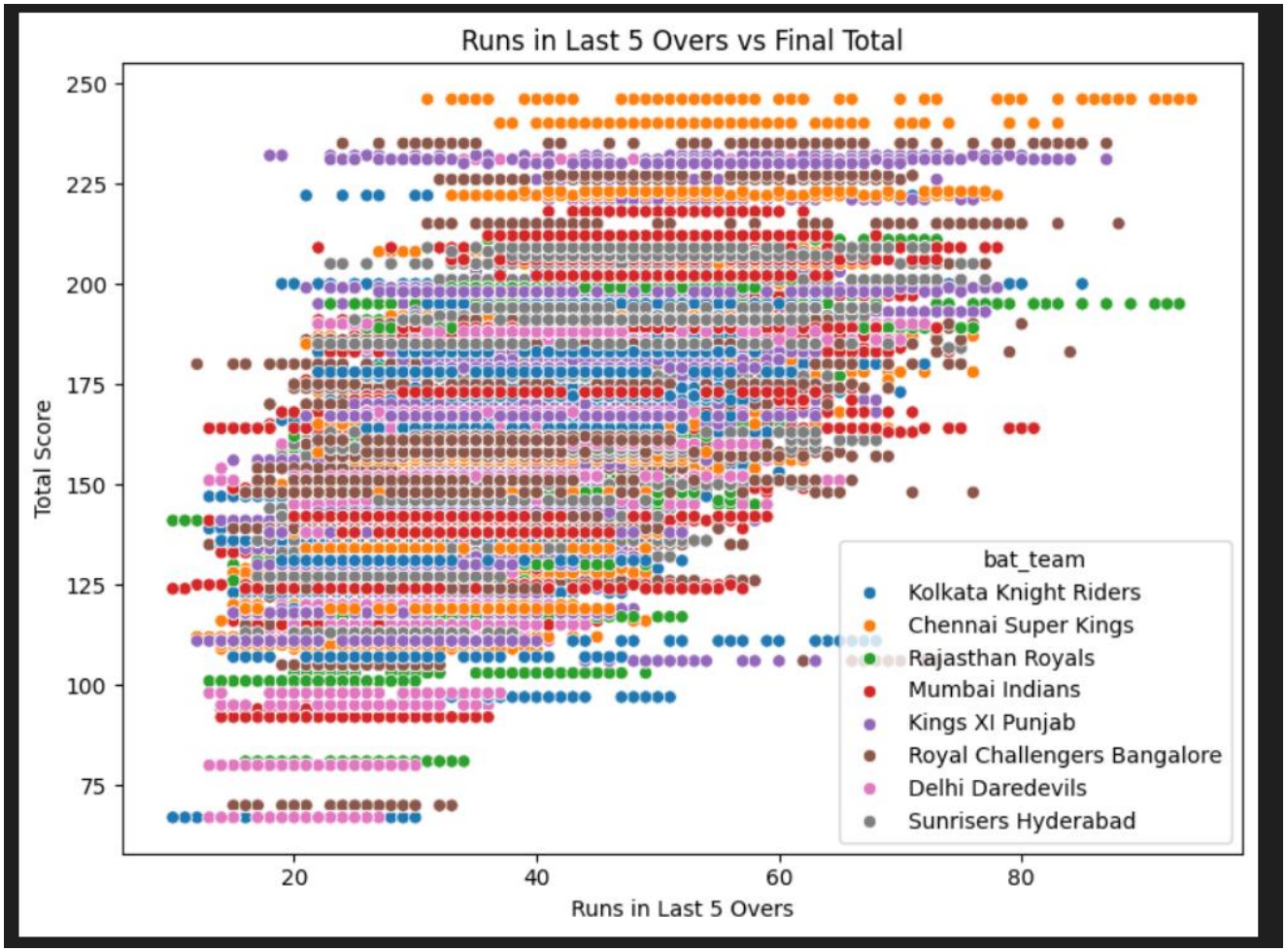
```
Before converting 'date' column from string to datetime object: <class 'str'>
After converting 'date' column from string to datetime object: <class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

```
    encoded_df = pd.get_dummies(data=df, columns=['bat_team', 'bowl_team'])
    encoded_df.columns
```

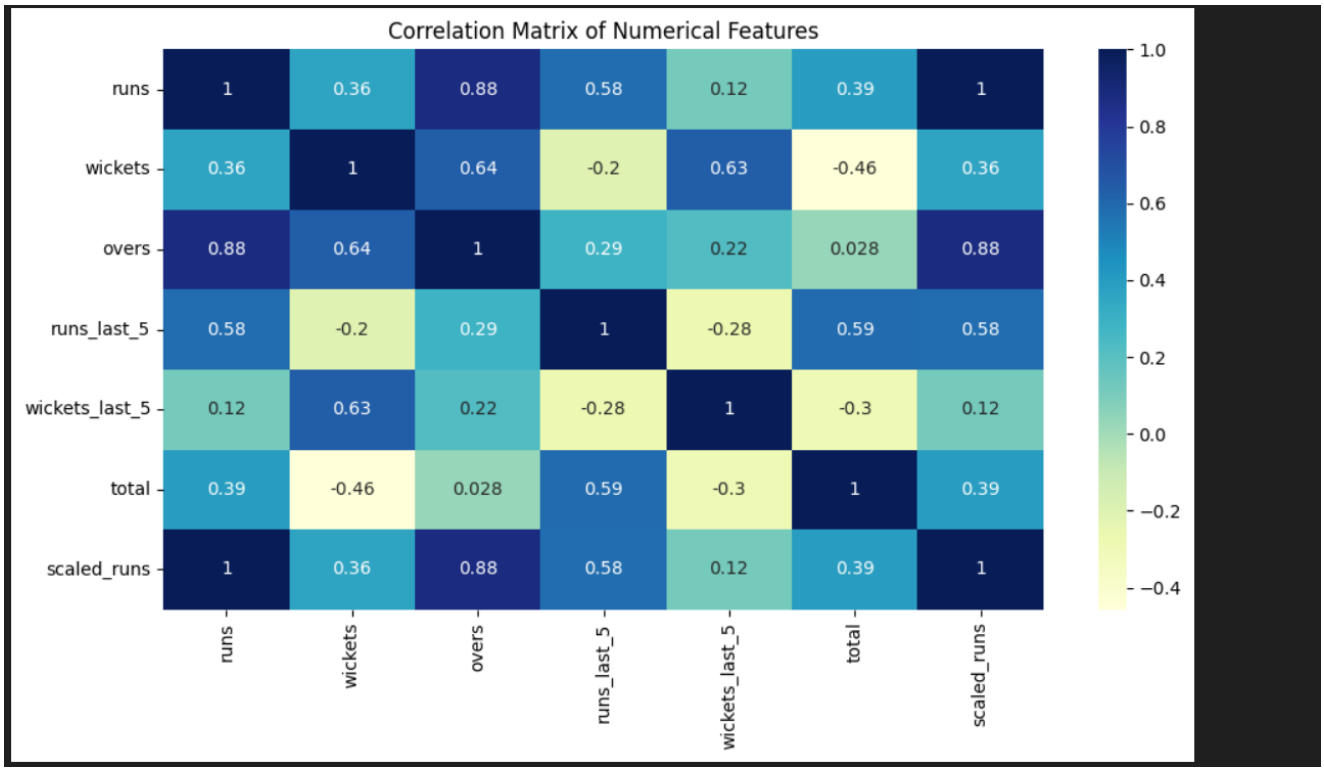
Python

```
Index(['date', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5',
       'total', 'scaled_runs', 'bat_team_Chennai Super Kings',
       'bat_team_Delhi Daredevils', 'bat_team_Kings XI Punjab',
       'bat_team_Kolkata Knight Riders', 'bat_team_Mumbai Indians',
       'bat_team_Rajasthan Royals', 'bat_team_Royal Challengers Bangalore',
       'bat_team_Sunrisers Hyderabad', 'bowl_team_Chennai Super Kings',
       'bowl_team_Delhi Daredevils', 'bowl_team_Kings XI Punjab',
       'bowl_team_Kolkata Knight Riders', 'bowl_team_Mumbai Indians',
       'bowl_team_Rajasthan Royals', 'bowl_team_Royal Challengers Bangalore',
       'bowl_team_Sunrisers Hyderabad'],
      dtype='object')
```

```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='runs_last_5', y='total', hue='bat_team')
plt.title('Runs in Last 5 Overs vs Final Total')
plt.xlabel('Runs in Last 5 Overs')
plt.ylabel('Total Score')
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='YlGnBu')
plt.title('Correlation Matrix of Numerical Features')
plt.tight_layout()
plt.show()
```



**1. Model Setup:**

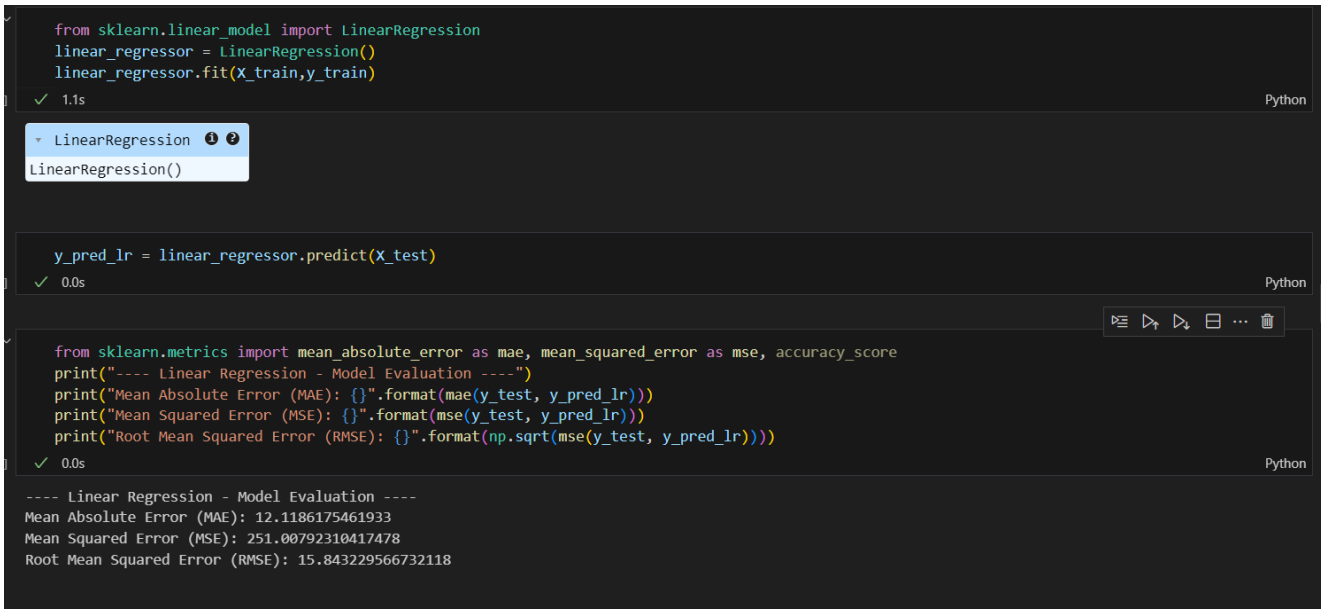
- The target variable is total_score.
- Features include *current score, wickets, overs, batting team, bowling team*, etc.

**2. Data Splitting:**

- The data is split into **training and test sets** using train_test_split.
- Ensures generalization and avoids overfitting.

**3. Model Training:**

- **Linear Regression:** Used as a baseline model.
- **Random Forest Regressor:** Applied to capture non-linear patterns and interactions.
- **Decision Tree Regressor:** Used for rule-based prediction and interpretability.





**1. Performance Metrics:**

- **$R^2$ Score:** Measures how well predicted scores match the actual ones.
- **MAE & RMSE:** Evaluate prediction error magnitudes.

- Metrics are calculated for each model for comparison.

**2. Visual Comparison:**

- **Scatter plots** show predicted vs. actual scores.
- **Bar plots** compare RMSE/$R^2$ scores across models using matplotlib.

```
from sklearn.ensemble import RandomForestRegressor
random_regressor = RandomForestRegressor()
random_regressor.fit(X_train,y_train)
```

30] ✓ 16.7s Python

```
RandomForestRegressor
RandomForestRegressor()
```

```
y_pred_rf = random_regressor.predict(X_test)
```

31] ✓ 0.0s Python

```
print("---- Random Forest Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_rf)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_rf)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_rf))))
```

32] ✓ 0.0s Python

```
---- Random Forest Regression - Model Evaluation ----
Mean Absolute Error (MAE): 13.710592555452706
Mean Squared Error (MSE): 330.1773682275355
Root Mean Squared Error (RMSE): 18.170783368570973
```

```
def predict_score(batting_team='Chennai Super Kings', bowling_team='Mumbai Indians', overs=5.1, runs=50, wickets=0, runs_in_prev_5=50, wickets_in_prev_5=0):
    temp_array = list()
    if batting_team == 'Chennai Super Kings':
        temp_array = temp_array + [1,0,0,0,0,0,0]
    elif batting_team == 'Delhi Daredevils':
        temp_array = temp_array + [0,1,0,0,0,0,0]
    elif batting_team == 'Kings XI Punjab':
        temp_array = temp_array + [0,0,1,0,0,0,0]
    elif batting_team == 'Kolkata Knight Riders':
        temp_array = temp_array + [0,0,0,1,0,0,0]
    elif batting_team == 'Mumbai Indians':
        temp_array = temp_array + [0,0,0,0,1,0,0]
    elif batting_team == 'Rajasthan Royals':
        temp_array = temp_array + [0,0,0,0,0,1,0]
    elif batting_team == 'Royal Challengers Bangalore':
        temp_array = temp_array + [0,0,0,0,0,0,1]
    elif batting_team == 'Sunrisers Hyderabad':
        temp_array = temp_array + [0,0,0,0,0,0,0]

    if bowling_team == 'Chennai Super Kings':
        temp_array = temp_array + [1,0,0,0,0,0,0]
    elif bowling_team == 'Delhi Daredevils':
        temp_array = temp_array + [0,1,0,0,0,0,0]
    elif bowling_team == 'Kings XI Punjab':
        temp_array = temp_array + [0,0,1,0,0,0,0]
    elif bowling_team == 'Kolkata Knight Riders':
        temp_array = temp_array + [0,0,0,1,0,0,0]
    elif bowling_team == 'Mumbai Indians':
        temp_array = temp_array + [0,0,0,0,1,0,0]
    elif bowling_team == 'Rajasthan Royals':
        temp_array = temp_array + [0,0,0,0,0,1,0]
    elif bowling_team == 'Royal Challengers Bangalore':
        temp_array = temp_array + [0,0,0,0,0,0,1]
    elif bowling_team == 'Sunrisers Hyderabad':
        temp_array = temp_array + [0,0,0,0,0,0,0]

    temp_array = temp_array + [overs, runs, wickets, runs_in_prev_5, wickets_in_prev_5]
    temp_array = np.array([temp_array])
    return int(linear_regressor.predict(temp_array)[0])
```

[33] ✓ 0.0s Python

Prediction 1
- Date: 16th April 2018
- IPL : Season 11
- Match number: 13
- Teams: Kolkata Knight Riders vs. Delhi Daredevils
- First Innings final score: 200/9

```
final_score = predict_score(batting_team='Kolkata Knight Riders', bowling_team='Delhi Daredevils', overs=9.2, runs=79, wickets=2, runs_in_prev_5=60, wickets_in_prev_5=1)
print("The final predicted score (range): {} to {}".format(final_score-10, final_score+5))
```
[38] Python

```
The final predicted score (range): 159 to 174
```

Prediction 2
- Date: 7th May 2018
- IPL : Season 11
- Match number: 39
- Teams: Sunrisers Hyderabad vs. Royal Challengers Bangalore
- First Innings final score: 146/10

```
final_score = predict_score(batting_team='Sunrisers Hyderabad', bowling_team='Royal Challengers Bangalore', overs=10.5, runs=67, wickets=3, runs_in_prev_5=29, wickets_in_prev_5=1)
print("The final predicted score (range): {} to {}".format(final_score-10, final_score+5))
```
[39] Python

```
The final predicted score (range): 138 to 153
```

Prediction 3
- Date: 17th May 2018
- IPL : Season 11
- Match number: 50
- Teams: Mumbai Indians vs. Kings XI Punjab
- First Innings final score: 186/8

```
final_score = predict_score(batting_team='Mumbai Indians', bowling_team='Kings XI Punjab', overs=14.1, runs=136, wickets=4, runs_in_prev_5=50, wickets_in_prev_5=0)
print("The final predicted score (range): {} to {}".format(final_score-10, final_score+5))
```
[40] Python

```
The final predicted score (range): 180 to 195
```

Prediction 4
- Date: 30th March 2019
- IPL : Season 12
- Match number: 9
- Teams: Mumbai Indians vs. Kings XI Punjab
- First Innings final score: 176/7

```
final_score = predict_score(batting_team='Mumbai Indians', bowling_team='Kings XI Punjab', overs=12.3, runs=113, wickets=2, runs_in_prev_5=55, wickets_in_prev_5=0)
print("The final predicted score (range): {} to {}".format(final_score-10, final_score+5))
```
[41] Python

```
The final predicted score (range): 179 to 194
```

Prediction 5
- Date: 11th April 2019
- IPL : Season 12
- Match number: 25
- Teams: Rajasthan Royals vs. Chennai Super Kings
- First Innings final score: 151/7

```
final_score = predict_score(batting_team='Rajasthan Royals', bowling_team='Chennai Super Kings', overs=13.3, runs=92, wickets=5, runs_in_prev_5=27, wickets_in_prev_5=2)
print("The final predicted score (range): {} to {}".format(final_score-10, final_score+5))
```
[42]

```
The final predicted score (range): 128 to 143
```

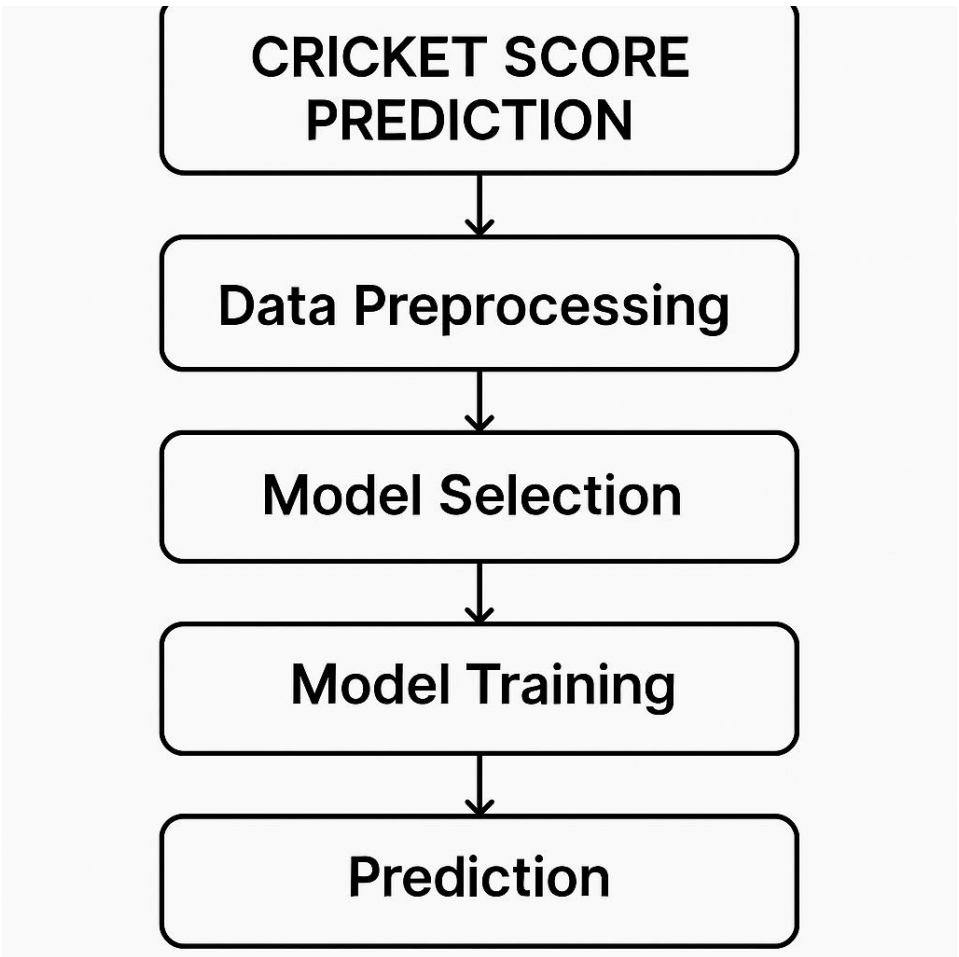
Prediction 6
- Date: 14th April 2019
- IPL : Season 12
- Match number: 30
- Teams: Sunrisers Hyderabad vs. Delhi Daredevils
- First Innings final score: 155/7

```
final_score = predict_score(batting_team='Delhi Daredevils', bowling_team='Sunrisers Hyderabad', overs=11.5, runs=98, wickets=3, runs_in_prev_5=41, wickets_in_prev_5=1)
print("The final predicted score (range): {} to {}".format(final_score-10, final_score+5))
```
[43]

```
The final predicted score (range): 157 to 172
```

```python
def predict_score_dynamic():
  batting_team = input("Enter the batting team: ")
  bowling_team = input("Enter the bowling team: ")
  overs = float(input("Enter the number of overs: "))
  runs = int(input("Enter the current runs: "))
  wickets = int(input("Enter the current wickets: "))
  runs_in_prev_5 = int(input("Enter runs scored in the last 5 overs: "))
  wickets_in_prev_5 = int(input("Enter wickets lost in the last 5 overs: "))
  temp_array = list()
  teams = ['Chennai Super Kings', 'Delhi Daredevils', 'Kings XI Punjab',
           'Kolkata Knight Riders', 'Mumbai Indians', 'Rajasthan Royals',
           'Royal Challengers Bangalore', 'Sunrisers Hyderabad']
  batting_team_index = teams.index(batting_team)
  batting_team_encoding = [0] * len(teams)
  batting_team_encoding[batting_team_index] = 1
  temp_array.extend(batting_team_encoding)
  bowling_team_index = teams.index(bowling_team)
  bowling_team_encoding = [0] * len(teams)
  bowling_team_encoding[bowling_team_index] = 1
  temp_array.extend(bowling_team_encoding)
  temp_array.extend([overs, runs, wickets, runs_in_prev_5, wickets_in_prev_5])
  temp_array = np.array([temp_array])
  predicted_score = int(linear_regressor.predict(temp_array)[0])
  print("The final predicted score (range): {} to {}".format(predicted_score - 10, predicted_score + 5))
predict_score_dynamic()
```

```
Enter the batting team: Chennai Super Kings
Enter the bowling team: Delhi Daredevils
Enter the number of overs: 15
Enter the current runs: 130
Enter the current wickets: 3
Enter runs scored in the last 5 overs: 30
Enter wickets lost in the last 5 overs: 1
The final predicted score (range): 170 to 185
```



## 5.CONCLUSION

This project successfully developed a cricket score prediction system using Machine Learning techniques. The workflow involved data preprocessing, feature selection, model training, performance evaluation, and visualization.

Among the various models applied, the Random Forest Regressor (or your chosen best model) demonstrated the highest accuracy and consistency in predicting match scores based on historical IPL data.

**Key Achievements:**

1. **EffectiveFeatureEngineering:**

   -- Extracted and utilized impactful features like venue, toss decision, team combination, and innings progression for accurate predictions.

2. **AccuratePredictions:**

   -- Achieved reliable first-innings and mid-match score forecasts, enhancing the potential for strategic insights.

3. **Visualization Dashboard:**

   -- Created clear and interactive plots to understand score trends and team performance.

**Challenges & Learnings:**

- **DataVariability:**

  Inconsistencies in player performance and weather conditions introduced unpredictability, demanding careful data curation.

- **ModelGeneralization:**

  Ensuring the model performs well across different seasons and teams required repeated validation and tuning.

**Future Enhancements:**

- Integrate live match data streams for real-time prediction updates.
- Include player-specific statistics (form, fitness, match-ups) to improve model depth.
- Deploy the model through a web or mobile app for broader accessibility to analysts and fans.