***A PROJECT ON***

# “CRICKET SCORE PREDICTION”

SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE COURSE OF

DIPLOMA IN BIG DATA ANALYSIS



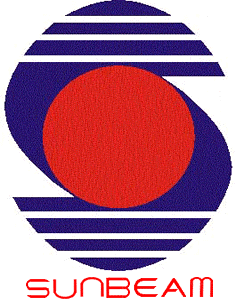
**SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, PUNE**

Submitted By:

Mrunal Chavhan (86988)

Payal Talmale (86827)

**Mr.Nitin Kudale Mrs.Manisha Hingne** Centre Coordinator Course Coordinator



**CERTIFICATE**

This is to certify that the project work under the title ‘Cricket Score Prediction’ is done by Mrunal Chavhan & Payal Talmale in partial fulfillment of the requirement for award of Diploma in Big Data Analysis Course.

Mr. Aniket P Mrs. Manisha Hingne

**Project Guide** **Course Coordinator**

Date:

# ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mrs. Manisha Hingne (Course Coordinator, SIIT ,Pune) and Project Guide Mr. Aniket P.

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Mrunal Chavhan

DBDA Aug 2024 Batch, SIIT Pune

Payal Talmale

DBDA Aug 2024 Batch, SIIT Pune

.

**TABLE OF CONTENTS**

1. **Introduction**
   1. Introduction And Objectives
   2. Why this problem needs To be Solved?
   3. Dataset Information

## Problem Definition and Algorithm

* 1. Problem Definition
  2. Algorithm Definition

## Experimental Evaluation

* 1. Methodology/Model
  2. Exploratory Data Analysis

## Results And Discussion

1. **GUI**
2. **GitHub link**

## 7.Future Work And Conclusion

* 1. Future Work
  2. Conclusion
     1. **Introduction**
        1. **Introduction And Objectives:**

Cricket is one of the most popular sports worldwide, with the T20 format being particularly fast-paced and unpredictable. Predicting the innings score in a match is crucial for teams, analysts, and fans, as it provides valuable insights into game strategy and match outcomes.

In this project, we focus on building a **Cricket Score Prediction Model** for **T20 IPL matches**, using **machine learning techniques**. We have collected and analyzed **historical match data from 2008 to 2024**, sourced from **Cricbuzz and Cricinfo**, and used **Selenium** for data scraping.

Our dataset includes key match features such as:

* **Batting Team & Bowling Team**
* **Venue** (categorized into Batting-friendly, Bowling-friendly, and Average)
* **Match Date & Season**
* **Current Over, Current Score, and Current Wickets**
* **Calculated Run Rate**

The **primary objective** of this project is to **predict the total innings score** based on real-time match conditions. We aim to develop an accurate **machine learning model** that can assist in:

* **Strategic Decision-Making**: Helping teams adjust their gameplay based on expected scores.
* **Fan Engagement**: Providing real-time insights to enhance the viewing experience.
* **Fantasy Leagues & Betting Analytics**: Aiding users in making data-driven decisions.
* **Match Performance Analysis**: Offering insights into team strategies and venue impact.

Using advanced **machine learning models**, including **XGBoost, Random Forest, and Regression-based models**, we optimize our predictions to achieve the highest accuracy. Our final model, fine-tuned using **Grid Search CV and Random Search Optimization**, achieves an **R² score of 93%**, making it a reliable tool for cricket analytics.

## Why this problem needs To be Solved?

Cricket score prediction is essential due to the dynamic nature of the game. Several key factors make this problem worth solving:

✅ **Optimizing Team Strategies**: Teams can adjust their batting and bowling approach based on predicted scores, improving decision-making in real-time.

✅ **Enhancing Fan Engagement**: Providing live score predictions increases excitement for fans, commentators, and broadcasters, making matches more interactive.

✅ **Impact on Marketing & Sponsorships**: Cricket franchises and brands can make data-driven sponsorship and advertisement decisions based on game trends.

✅ **Better Inventory & Resource Management**: Predicting match scores can help stadium management prepare for crowd engagement, refreshments, and logistics.

✅ **Sports Betting & Fantasy Leagues**: Accurate predictions assist users in fantasy leagues and betting platforms to make informed choices.

By applying **data science and machine learning**, we aim to **bridge the gap between cricket analytics and real-time decision-making**, revolutionizing the way match performances are analyzed and predicted.

.

## Dataset Information.

For this project, we have compiled and merged data from two sources: **Cricbuzz and Cricinfo**, covering IPL matches from **2008 to 2024**.

* **Combined-Crickinfo Data (2008-2016)**: **21,459 rows**
* **Cricbuzz Data (2017-2024)**: **19,066 rows**
* **Total Combined Dataset**: **40,525 rows**

**Columns :**

1. **Batting Team:** The team currently batting in the match.
2. **Bowling Team:** The team currently bowling in the match.
3. **Venue:** The stadium where the match is being played.
4. **Over Number:** The current over of the innings.
5. **Score after Over:** Cumulative score at the end of a specific over.
6. **Wickets after Over:** Total wickets lost at the end of a specific over.
7. **Inning Score:** The total score achieved by the batting team at the end of the innings.
8. **Year:** Extracted from the match date for trend analysis.
9. **Month:** Extracted from the match date for trend analysis.
10. **Day:** Extracted from the match date for trend analysis.
11. **Day\_Sin:** Cyclical encoding of the day of the month.
12. **Day\_Cos:** Cyclical encoding of the day of the month.
13. **Run Rate:** Current scoring rate, calculated as (current score / overs faced).
14. **Batting Home Ground:** Indicates if the batting team is playing at its home venue.
15. **Bowling Home Ground:** Indicates if the bowling team is playing at its home venue.
16. **Pitch Type:** Categorized as Batting-friendly, Bowling-friendly, or Average.
17. **Season:** Derived from the match year to group matches seasonally.

## Problem Definition and Algorithm:

The problem is straightforward: given historical T20 IPL match data, we aim to predict the final innings score of a team based on real-time match conditions. Our dataset includes match details such as the batting and bowling teams, venue, over number, current score, wickets, and run rate, along with additional calculated features.

The goal is to train a machine learning model that can accurately forecast the innings score. We evaluate model performance using **Mean Absolute Error (MAE)** and **R² score** as our key metrics. The closer our predicted score is to the actual innings total, the better the model's performance.

Since cricket matches are influenced by multiple factors like pitch conditions, venue history, and team performance, our model must capture these dynamics effectively to provide reliable predictions. The ultimate objective is to develop a system that enhances cricket analytics, aiding teams, analysts, and fans with real-time insights.

## Algorithm Definition

We experimented with multiple machine learning algorithms to predict the final innings score of a T20 IPL match. Below is an overview of the models used:

#### **Linear Regression**

#### **Linear Regression is a supervised learning algorithm used for predicting continuous values. It assumes a linear relationship between independent variables (features) and the dependent variable (target). The algorithm fits a straight line to the data by minimizing the difference between actual and predicted values.**

#### **How It Works:**

#### **The model assigns weights to each feature and finds the best-fitting line that minimizes prediction errors.**

#### **Predictions are made by summing up the weighted feature values.**

#### **The best-fit line is determined by minimizing the sum of squared errors between actual and predicted values.**

*Polynomial Regression*

Polynomial Regression extends Linear Regression by introducing higher-degree terms, allowing it to fit curved patterns in data.

How It Works:

Instead of fitting a straight line, it fits a curved function by adding polynomial terms.

Useful when the relationship between variables is non-linear but still continuous.

*Ridge Regression*

Ridge Regression is an extension of Linear Regression that adds a regularization term to prevent overfitting by shrinking large coefficient values.

How It Works:

It introduces a penalty on large feature coefficients to reduce model complexity.

This prevents excessive sensitivity to small variations in the training data, making the model more robust.

Suitable for datasets with highly correlated features (multicollinearity).

#### **Lasso Regression**

#### **Lasso Regression is similar to Ridge but with a stronger penalty that forces some feature coefficients to be completely zero, effectively performing automatic feature selection.**

#### **How It Works:**

#### **Adds a penalty term that shrinks unnecessary feature coefficients to zero.**

#### **This results in a sparse model, keeping only the most important features.**

#### **Helps in reducing model complexity and improving interpretability.XGBoost (Extreme Gradient Boosting)**

*Decision Tree Regressor*

A Decision Tree Regression model splits data into smaller regions based on feature values, making stepwise predictions instead of using a single function like Linear Regression.

How It Works:

The dataset is split at different points based on feature values to create a tree-like structure.

Each node represents a decision, and the final predictions are obtained from the leaf nodes.

The splits are chosen to minimize prediction errors at each step.

*Support Vector Regressor (SVR)*

SVR is an extension of Support Vector Machines (SVM) used for regression, aiming to find a hyperplane that fits most data points within a certain margin of error.

How It Works:

Unlike other regression models, SVR focuses on finding a function that predicts values within an acceptable error range.

Uses a kernel trick to handle non-linear relationships by mapping input features to a higher-dimensional space.

The model only penalizes predictions that fall outside a specified margin, making it robust to small fluctuations in data.

*Random Forest Regressor*

Random Forest is an ensemble learning technique that builds multiple Decision Trees and averages their predictions to improve accuracy and reduce overfitting.

How It Works:

The algorithm randomly selects subsets of data and features to train multiple Decision Trees.

Each tree makes a prediction, and the final output is the average of all predictions.

Since each tree sees different parts of the data, the model is less prone to overfitting than a single Decision Tree.

*XGBoost Regressor*

XGBoost (Extreme Gradient Boosting) is an advanced boosting algorithm that builds Decision Trees sequentially, improving performance at each step.

How It Works:

Instead of training all trees independently, XGBoost corrects the errors of previous trees by focusing more on misclassified data.

Uses gradient descent to optimize the error reduction.

Employs regularization to prevent overfitting, making it superior to standard Gradient Boosting.

*Gradient Boosting Regressor*

Gradient Boosting builds models sequentially, where each new model reduces the error of the previous model by learning from its mistakes.

How It Works:

The first model makes initial predictions.

The next models focus on reducing the errors of the previous models.

The final prediction is obtained by summing the contributions of all models.

*AdaBoost Regressor*

AdaBoost (Adaptive Boosting) builds weak models (Decision Trees with few splits) and improves them by giving more weight to incorrect predictions in subsequent iterations.

How It Works:

Initially, all samples have equal importance.

After each iteration, misclassified samples receive higher importance.

The final model combines all weak models, giving higher influence to those with better accuracy.

A powerful gradient boosting algorithm optimized for performance and efficiency. It sequentially improves predictions by correcting errors from previous iterations. XGBoost achieved the **highest accuracy (93% R² score)** after hyperparameter tuning.

We evaluated model performance using **Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R² score**, selecting **XGBoost** as the final model due to its superior accuracy and optimization capabilities.

## Experimental Evaluation:

* + - 1. **Methodology:**

The goal is to predict T20 innings scores using historical IPL match data from 2008 to 2024. The data is collected from Cricbuzz and Cricinfo and merged into a single dataset for analysis.

***Loading the data :***

df = pd.read\_csv("combined\_file.csv")

df1 = pd.read\_csv("combined\_crickinfo\_data.csv")

Two datasets are loaded: combined\_file.csv (19,066 rows) and combined\_crickinfo\_data.csv (21,459 rows).

Duplicates and irrelevant columns (Match Time, Inning Wickets) are removed.

***Preprocessing Steps:***

* Column Standardization: Venue column name is corrected.
* Date Formatting:

Match dates are cleaned to remove trailing commas and extra text.

Date ranges (e.g., May 27 - 28 2014) are converted to a single valid date.

The Match Date column is transformed into a standard datetime format.

df1['Match Date'] = df1['Match Date'].str.replace(r'[,\\n].\*', '', regex=True)

df1['Match Date'] = df1['Match Date'].str.replace(r'-.\*', '', regex=True)

df1['Match Date'] = df1['Match Date'].str.strip()

df1['Match Date'] = pd.to\_datetime(df1['Match Date'], format='%B %d %Y', errors='coerce')

***Feature Engineering*:**

The dataset includes additional calculated columns such as Run Rate, Day\_Sin, Day\_Cos, and Season.

The venues are categorized into three pitch types: Batting-friendly, Bowling-friendly, and Average.

***Handling Missing Values :***

Any missing values in relevant columns are filled using appropriate strategies:

Mode/Mean for Numerical Data

Categorical Imputation for Categorical Data

## Flow Diagram :

## 

### **3.2 Exploratory Data Analysis (EDA)**

Heatmap Analysis of Maximum Inning Scores Across Venues and Years

This **heatmap visualization** represents the **maximum innings scores recorded at various venues over different years**, based on the selected **batting and bowling teams**. The **x-axis represents the years (2008–2024)**, while the **y-axis lists different cricket venues** where matches were played. The color intensity indicates the **maximum inning score**, with darker shades representing higher scores.



**Fig 1:** **Venue vs Inning Score (Filtered by Batting & Bowling Teams) (heat map)**

Key Observations:

High-Scoring Venues: Arun Jaitley Stadium (2024) recorded the highest score (219), followed by MA Chidambaram Stadium (2008) with 202.

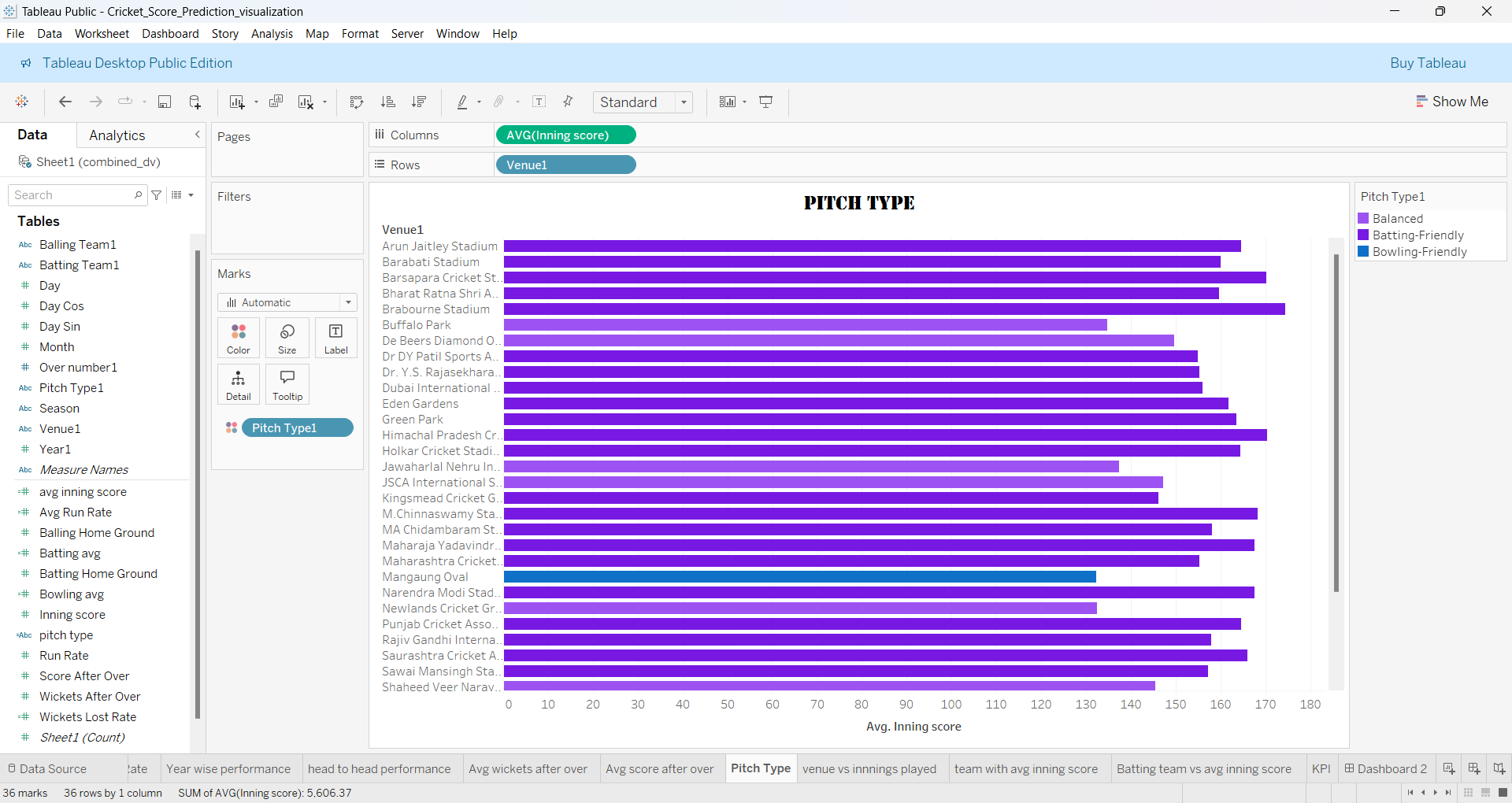
Consistent High Scores: Wankhede Stadium has frequently seen high scores, indicating a batting-friendly pitch.

Score Trends Over Years: Some venues have missing data for certain years, suggesting no matches or low-scoring games.

Bowling Impact: Certain teams have restricted scores at specific venues, highlighting strong bowling performances.

Venue-Based Inning Score Analysis: Impact of Pitch Conditions on Scoring Trends

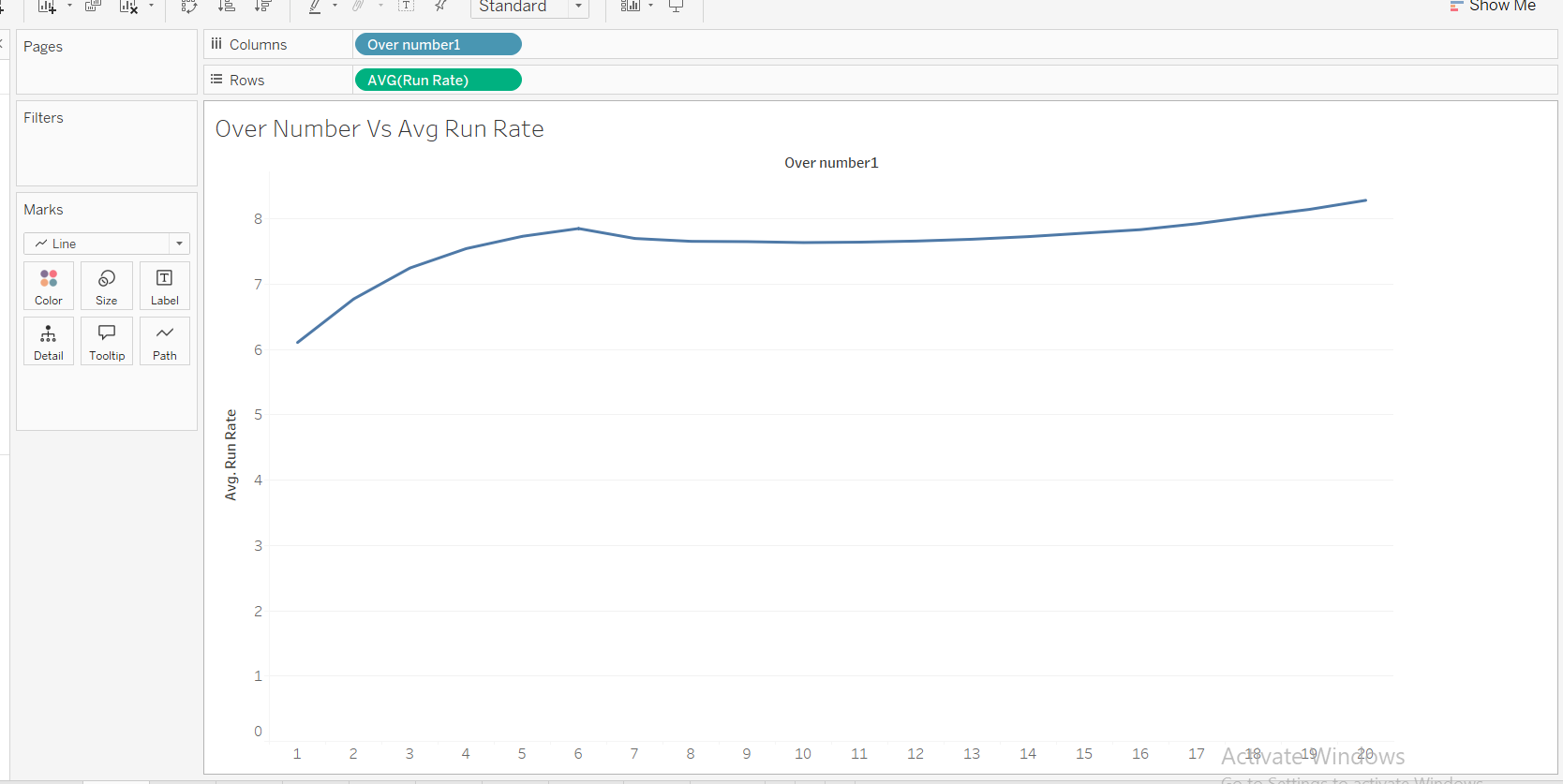
The bar chart provides an analysis of average inning scores across various cricket venues, categorized by pitch type. The pitch types include **Balanced**, **Batting-Friendly**, and **Bowling-Friendly**, with Balanced pitches (purple) being the most common. The *Bowling-Friendly* venue, *Mangaung Oval,* is distinctly marked in blue and has a relatively lower average inning score. Most venues have high average inning scores, indicating *batting-friendly* conditions. A few venues show moderate scores, suggesting bowler-friendly factors like pitch behavior and weather conditions. This analysis helps in understanding venue-based performance trends, which can assist in match strategies and score prediction models.



**Fig 2. Pitch Type**

Run Rate Progression Over Innings in T20 Matches

The line chart visualizes the average run rate across different overs in a T20 innings. The x-axis represents the over number (1 to 20), while the y-axis shows the corresponding average run rate.



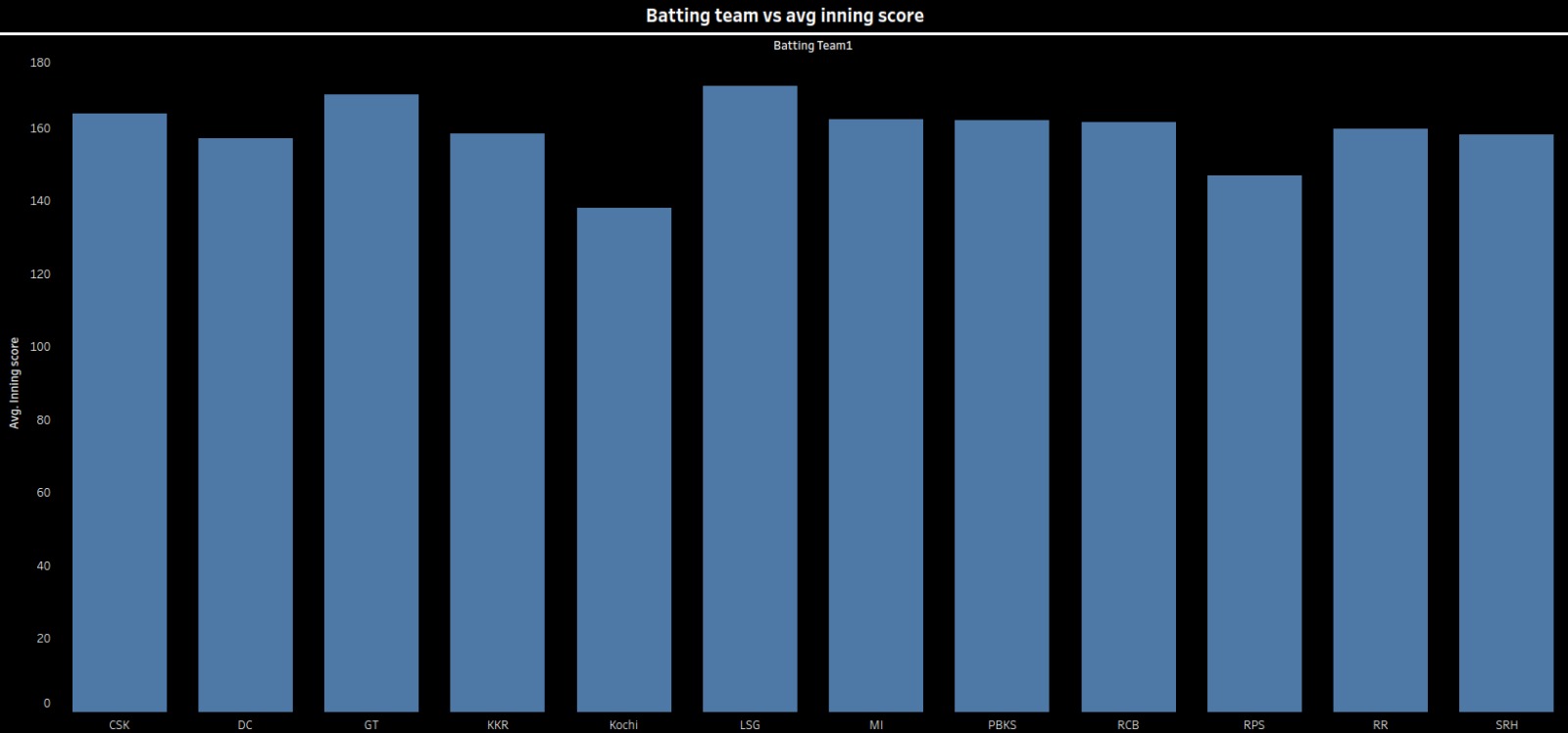
**Fig 3: Run Rate Progression Chart**

Key Observations:

1. The run rate starts relatively low in the initial overs (~6 RPO) as teams focus on building a foundation.
2. There is a steady increase in the run rate, peaking around the 7th or 8th over (~8 RPO), indicating aggressive batting in the middle overs.
3. A slight dip follows, suggesting a more cautious approach or effective bowling during the middle phase.
4. The run rate picks up again towards the death overs (16-20), where teams look to accelerate and maximize scoring opportunities.
5. This analysis highlights the typical scoring pattern in T20 cricket, where teams start cautiously, build momentum in the middle, and finish aggressively.

Team-Wise Batting Performance: Average Inning Scores in IPL

This bar chart displays the average inning scores for different IPL teams when batting. Each bar represents a specific team, and the height of the bar indicates its average score.

**Fig 4 : Bar Chart – Batting Team vs. Average Inning Score**

Key observations:

Some teams, like MI and LSG, have higher average scores, indicating strong batting performances.

Teams like Kochi and KKR show comparatively lower averages, suggesting possible inconsistencies or weaker batting performances.

The variation in scores across teams highlights the impact of batting strength, pitch conditions, and opposition bowling attacks.

## Results and discussion:

In this project, various machine learning algorithms were applied to predict the final score of T20 cricket matches, based on historical match data scraped from Cricbuzz. The following algorithms were tested:

Linear Regression

Lasso Regression

Ridge Regression

Random Forest

Decision Tree

Gradient Boosting Machine (GBM)

Among these, the XGBoost Machine Learning algorithm outperformed the others, providing the highest R² score of 0.93. This indicates that the XGBoost model was able to predict the match scores with high accuracy.

from xgboost import XGBRegressor

from sklearn.model\_selection import RandomizedSearchCV

import numpy as np

# Define the XGBoost model

xgb\_model = XGBRegressor(objective='reg:squarederror', random\_state=42)

# Define the hyperparameter space

param\_dist = {

'learning\_rate': np.linspace(0.01, 0.3, 5), # 0.01 to 0.3 in 5 steps

'max\_depth': [3, 5, 7, 9, 11],

'subsample': [0.6, 0.8, 1.0],

'colsample\_bytree': [0.6, 0.8, 1.0],

'n\_estimators': [50, 100, 300, 500, 700],

'gamma': [0, 0.1, 0.3, 0.5, 1.0],

'reg\_lambda': [0.01, 0.1, 1, 10], # L2 Regularization

'reg\_alpha': [0.01, 0.1, 1, 10] # L1 Regularization

}

# Set up RandomizedSearchCV

random\_search = RandomizedSearchCV(

estimator=xgb\_model,

param\_distributions=param\_dist,

n\_iter=30, # Number of random combinations to test

scoring='r2',

cv=5, # 5-fold cross-validation

verbose=3,

n\_jobs=-1, # Use all available processors

random\_state=42

)

# Fit RandomizedSearchCV to the data

random\_search.fit(X\_train, y\_train)

# Best parameters and score

print("Best Parameters:", random\_search.best\_params\_)

print("Best R2 Score:", random\_search.best\_score\_)

# Use the best model

best\_xgb = random\_search.best\_estimator\_

Fitting 5 folds for each of 30 candidates, totalling 150 fits

Best Parameters: {'subsample': 0.8, 'reg\_lambda': 1, 'reg\_alpha': 0.1, 'n\_estimators': 700, 'max\_depth': 9, 'learning\_rate': 0.08249999999999999, 'gamma': 0.3, 'colsample\_bytree': 0.8}

Best R2 Score: 0.932990312576294

MAE : 1940.99

R2 Score : 0.94

## GUI:

GUI is made using Flask framework. **Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools

**6.GitHubLink:**

[**https://github.com/mrunal-chavhan/CDAC\_PROJECT.git**](https://github.com/mrunal-chavhan/CDAC_PROJECT.git)

**7.Future work and Conclusion**

## 7.1Future Work:

* Real-Time Data Processing: Integrating live match data from sources like Cricbuzz API or ESPN Cricinfo would allow for real-time predictions, making the model applicable during ongoing T20 matches, offering dynamic insights based on live conditions.
* Incorporating Additional Data Features: The accuracy of predictions can be improved by including additional features, such as:
* Player-specific data, such as individual batting, bowling averages, and recent form.
* Match-related statistics, like overs bowled, player injuries, and weather conditions, can also impact the game’s outcome and should be included in the feature set.
* Team-specific statistics, such as recent team form, venue-specific performance, and player availability, can also provide valuable insights into predicting match scores.

## 7.2 Conclusion:

In this project, machine learning algorithms were used to predict the final scores of T20 cricket matches. Among the models tested, Gradient Boosting Machine (GBM) and XGBoost provided the highest accuracy in predicting the final scores, with an R² score of 0.93 respectively. These models outperformed traditional regression models like Linear Regression and Decision Trees, showcasing the power of ensemble methods for predicting complex sports data.

The project successfully demonstrated the effectiveness of historical match data in predicting future match outcomes, with the XGBoost model standing out as the most robust performer. Additionally, the model's feature importance analysis provided valuable insights into the factors that most influence match scores, such as team performance, wickets, and venue conditions.

Future work will focus on improving the model through hyperparameter optimization, incorporating more features, and exploring real-time data integration to enhance its predictive capabilities. By refining the model and adding new data sources, it can be further adapted to predict not only scores but also other match outcomes, benefiting analysts, teams, and enthusiasts in the world of T20 cricket.