Unveiling Cricket Dynamics: Extracting Insights from Ball-By-Ball Data

By: Pragnya Vijayan and Vignesh Senthilkumar

Background:

I. Setting

Cricket stands as one of the globe's most beloved sports, drawing in a massive audience. The Men's World Cup of 2023 alone witnessed an astonishing 16.9 billion viewers tuning in. A truncated version of cricket known as Twenty20, or T20, has surged in popularity due to its fast-paced and action-filled gameplay. This interest has propelled the global cricket market to a whopping $289.89 million. Given the intense nature of the sport, using past matches to gain insights for future games is crucial. This retrospective analysis can shed light on performance dynamics, aid in strategic player selection based on match conditions, and facilitate performance comparison using metrics such as economy and strike rate.

This study will concentrate on the 2022 T20 Asia Cup, where each inning is restricted to 20 overs per team. An over consists of six deliveries pitched by the bowler to the opposing team's batsman. The batsman aims to accrue runs for their team, while the opposing team strives to dismiss them by taking their wicket.

Although this study began with the hope of extracting crucial information from the live commentary of the game, the analysis of the multilayer perceptron model could not be completed due to a lack of GPU resources (see more in the last section). The accompanying code is attached to the paper. Thereby, this research will focus on getting and comparing team and over-based statistics using data visualization.

II. Data Source

The Asia Cup tournament data was gathered by conducting web scraping from the ESPN Cricinfo website and subsequently made accessible for use on Kaggle.

Design:

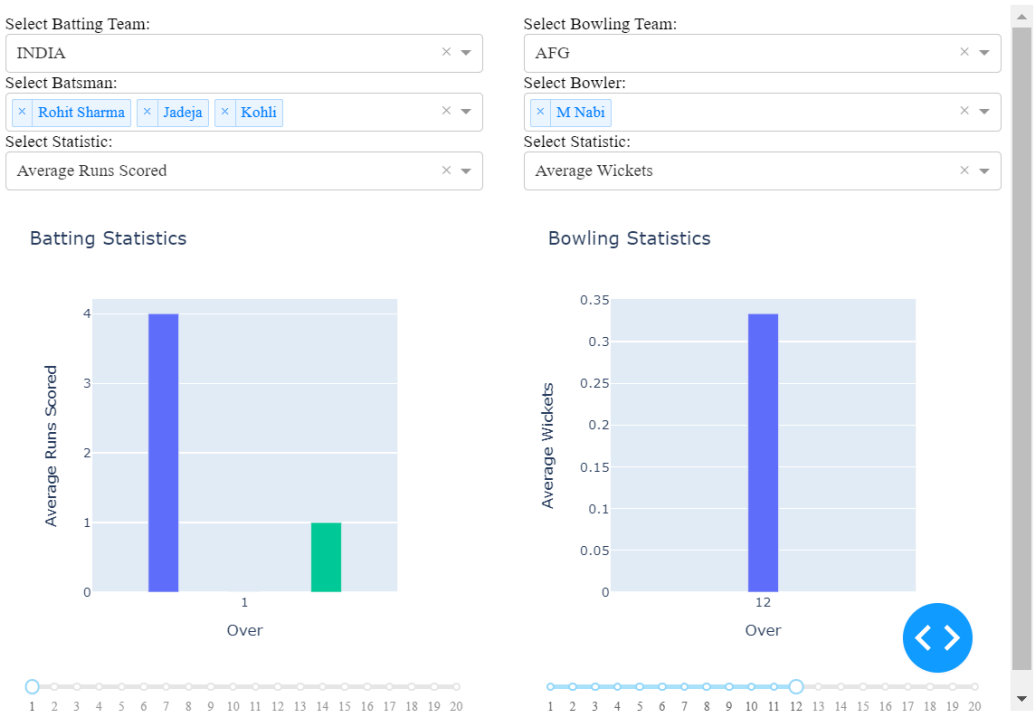
I. Preprocessing

A. The dataset lacked crucial feature columns like bowling team and winning team. To address this, the bowling team was identified by referencing the inning number and determining the team that hadn't played in the opposite inning. Meanwhile, the winning team information was obtained from the Cricinfo website and manually updated in the dataset.

B. The runs column in the dataset had tags like W and NB along with the numerical runs. Used regex to create separate columns for the numerical runs and extras.

II. Data Visualization

A. The implementation of an interactive graphing functionality using Dash, a Python framework for web apps, was crucial for facilitating the comparison of player statistics both within and across teams, as well as visualizing match outcomes effectively. This approach prioritized the clarity and accessibility of data visualization, ensuring that users could easily analyze and interpret cricket match data for informed insights.



Example usage of the dash web app. To enhance usability, a slider element was incorporated into the graph, enabling users to toggle between overs, while dropdown menus provided options for selecting teams and players.

III. Targets

A. Our code aims to predict the performance of cricket batsmen and bowlers. We've chosen to focus on two primary statistics that are fundamental to understanding player proficiency in cricket: runs for batsmen and wickets for bowlers. These statistics serve as important indicators of player effectiveness and contribution to their teams' success.

B. Linear regression was chosen for its interpretability, ease of implementation, and suitability as a realistic model in sports analytics. Its straightforward interpretation of feature coefficients helps provide for a complete understanding of the listed features' impact on the target variable.

Implementation:

A. To calculate the statistics for bowlers and batsman, our code analyzes the data to derive insightful statistics for both bowlers and batsmen. It initializes dictionaries for each

statistical aspect, including runs conceded, wickets taken, balls bowled, economy rate, and runs distribution for bowlers, as well as total runs scored, runs distribution, dismissals, innings played, balls faced, and strike rate for batsmen.

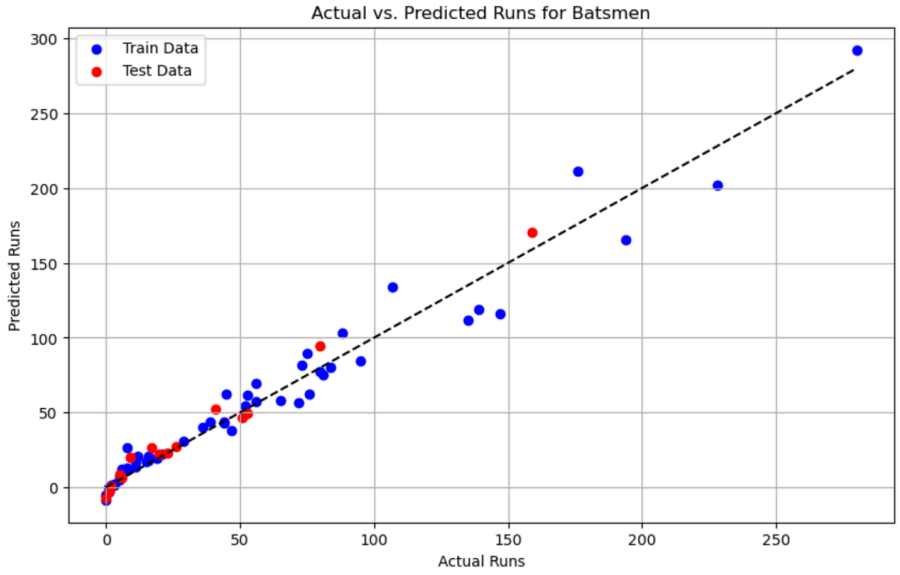
B. By iterating through the match data, our code updates these dictionaries, incrementing counts and computing key metrics such as runs conceded, wickets taken, economy rate, total runs scored, dismissals, and strike rate. Additionally, it categorizes runs into different types and calculates the distribution of runs per bowler and per batsman, providing a detailed breakdown of the bowlers and batsman performance.
C. Our code for the predictive model incorporates a variety of techniques to analyze cricket player performance effectively. First, we used linear regression models to predict the performance of both batsmen and bowlers. Second, employed mean squared error (MSE) as a performance metric to evaluate the accuracy of our model from training and testing datasets. MSE provides a good quantitative measure of the average squared difference between actual and predicted values.

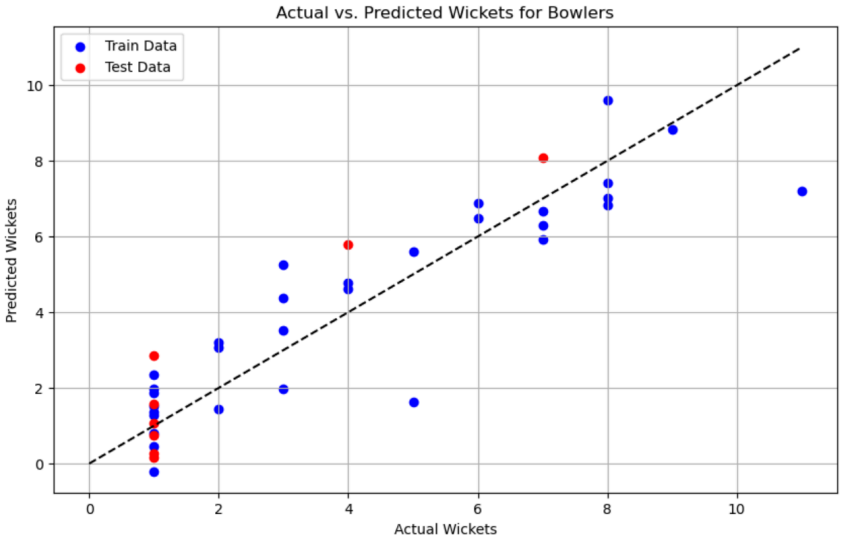
Results:

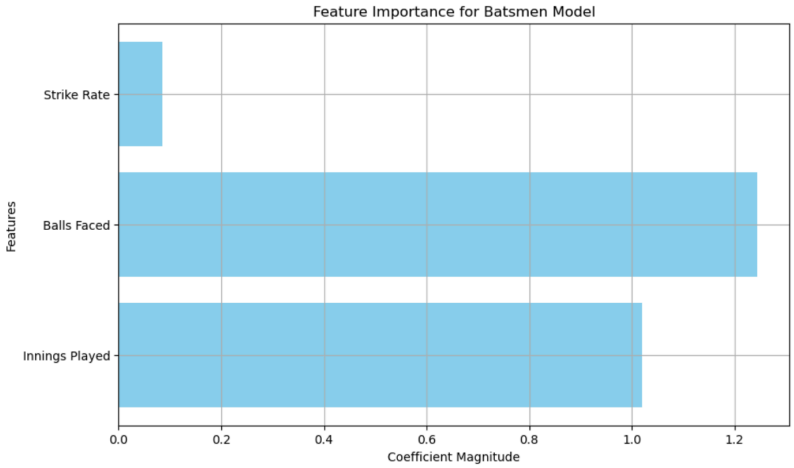
A. The final report presents quantitative results obtained through our dash app, showcasing batting and bowling statistics for cricket matches. We calculate average runs scored and strike rates per over for batsmen, and average wickets taken and runs conceded per over for bowlers, offering detailed insights into player performance throughout the match. Utilizing visualization tools, the report illustrates these findings through interactive plots. Moreover, the report highlights the capability to plot multiple batsmen or bowlers from each team on the same plot, enabling comparative analysis of player performance within a team context.
B. The final report integrates data analysis using visualization tools, employing interactive plots from the dash app and regression model to depict batting and bowling statistics dynamically. The evaluation of model learning uses mean squared error to assess its performance. Graphs within the report feature labeled scatter plots for actual vs. predicted runs/wickets and feature importance plots. Here are the figures:

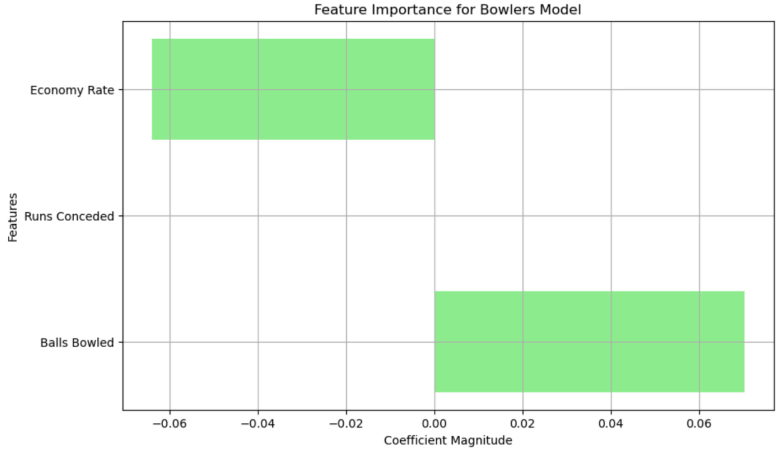
Model Output:

```
Batsmen Model - Train MSE: 142.34629857295224
Batsmen Model - Test MSE: 53.454720644195795
Bowlers Model - Train MSE: 1.6782345615009429
Bowlers Model - Test MSE: 1.1726819812443443

Top 5 Batsmen (Predicted):
1. Rizwan - Predicted Total Runs: 292.42
2. Ibrahim Zadran - Predicted Total Runs: 211.38
3. Kohli - Predicted Total Runs: 202.31
4. Nissanka - Predicted Total Runs: 170.24
5. Rajapaksa - Predicted Total Runs: 165.59

Top 5 Bowlers (Predicted):
1. Theekshana - Predicted Total Wickets: 9.61
2. Hasaranga de Silva - Predicted Total Wickets: 8.84
3. Madushanka - Predicted Total Wickets: 8.10
4. Haris Rauf - Predicted Total Wickets: 7.42
5. Bhuvneshwar - Predicted Total Wickets: 7.20
```




Actual vs. Predicted Wickets for Bowlers
Train Data
Test Data
Predicted Wickets
Actual Wickets
0
2
4
6
8
10

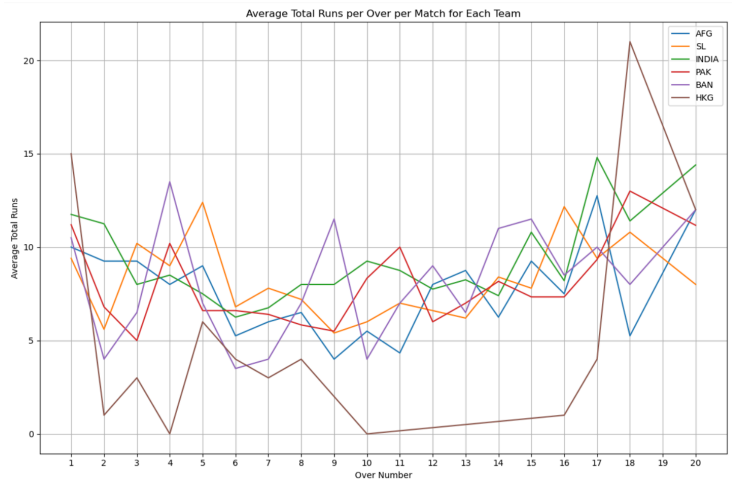

Feature Importance for Batsmen Model
Strike Rate
Balls Faced
Innings Played
Features
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Coefficient Magnitude

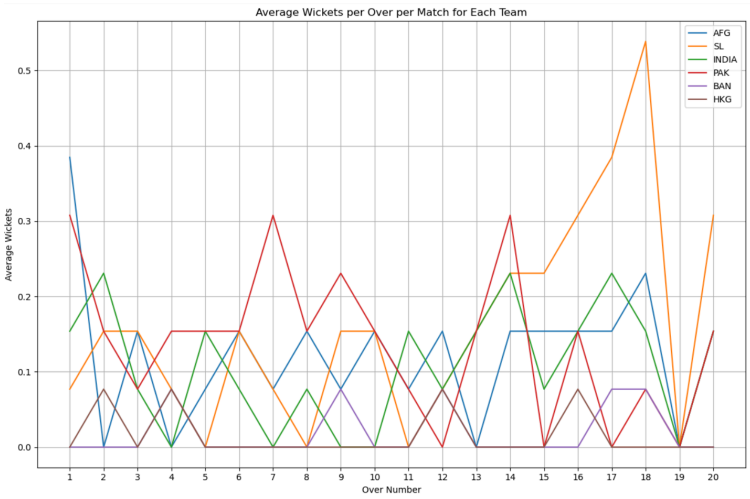

Feature Importance for Bowlers Model
Economy Rate
Runs Conceded
Balls Bowled
Features
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
Coefficient Magnitude

Interpretation:

A. Based on the results, we can conclude that our model is able to predict the best bowlers with a MSE of 1.17 and best batsmen with a MSE of 53.45, which tells us that the model effectively predicts the bowlers' wickets but there is more room for improvement for the batsmens' runs. We can see that the data points are relatively linear and grouped together for the actual v.s. predicted plots for lower total runs and lower total wickets and are more spread apart for higher total runs and higher total runs. We can also conclude that the number of balls faced and balls bowled have the most importance on the batsman and bowler models respectively.

Other Figures:





Natural Language Processing

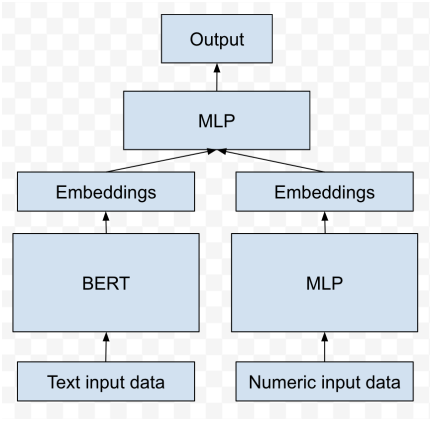
Background

Drawing inspiration from a previous study utilizing Natural Language Processing (NLP) data to predict match outcomes, the project aims to integrate contextual factors such as player injuries and weather conditions, often discussed in commentaries, to ensure a nuanced analysis. The study highlighted the potential of capturing on-field emotions and players' adaptability to field conditions and opponents, providing valuable insights for teams and players to optimize practice regimes and strategic planning for upcoming games and tournaments. Given that the best-performing model in the referenced study utilized BERT for tokenization and model fitting, the model aims to replicate this approach. However, several challenges arise in this analysis: firstly, the complexity of working with transformer models; secondly, the fact that BERT models are primarily tailored for text, whereas the

dataset encompasses textual, numerical, and categorical data; thirdly, the necessity of constructing a Multilayer Perceptron (MLP) model; and finally, the GPU computational resources required are inaccessible and not dependable.

Design

As mentioned, the dataset had textual, numerical, and categorical data that were necessary to be used as feature columns. However, BERT is primarily used for NLP and would not work properly with numerical data. With this in mind, several approaches were considered. The first is splitting the dataset into numerical and categorical data, running BERT tokenization and model fitting on the textual data while having the numerical data go through the MLP, and then concatenating both sets of embeddings, and sending the resultant embedding into another MLP model to get the output prediction (See figure). Another approach was to add context to the numerical data in the form of words, and then append it to textual commentary before processing it in a way a fully textual data would. The second approach was implemented due to the ease of having to implement only one MLP model (the left path taken in figure).



Implementation

I. Preprocessing

The preprocessing of the textual data involved a sequential pipeline consisting of the following steps:

- Converting the text to lowercase.
- Removing punctuation marks.
- Handling numerical data.
- Correcting spelling errors.
- Performing named entity recognition.
- Replacing certain elements in the text.
- Stemming words to their root forms.

- Lemmatizing words to their base forms.
- Removing stopwords from the text.

Some Eastern given names and country names that weren't previously removed were removed later using stopword removal.

II. New dataset

Rather than continue with the ball-by-ball format, the study pivoted to using an over-by-over format such that it can better predict the average number of runs scored in an over or the probability of losing a wicket in an over. Used the 'groupby' function to create the new dataset.

III. MLP Preparation

Data from all the necessary feature columns were combined, such that it was in a list of strings format for all the rows in the dataset. The label was the winning team. Then, the textual information was tokenized and created into an embedding. It was split into test, train, and validation sets, and made into tensors, which were then inputted to the MLP model.

Results

Due to challenges with using the GPU, the NLP portion of the study was unable to be completed. With more time, this part of the project could be completed.

Citation:

- Impact Calculation Of The Players Using The Cricket Commentary Corpus