National College of Ireland

Cricket Shot Prediction using Match Context Data by use of XGBoost Classifier

MSc Research Project
MSc Data Analytics

Megh Vijay Shirke
Student ID: x23302682

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Asif Razzaq

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Megh Vijay Shirke
Student ID:	x23302682
Programme:	MSc Data Analytics
Year:	2025-26
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Asif Razzaq
Submission Due Date:	11/12/2025
Project Title:	Cricket Shot Prediction using Match Context Data by use of XGBoost Classifier
Word Count:	9206
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Megh
Date:	11th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Cricket Shot Prediction using Match Context Data by use of XGBoost Classifier

Megh Vijay Shirke
x23302682

Abstract

A newer and popular use of sports analytics has been in batting shot prediction, namely in the context of cricket where real-time decision support can be applied to improve coaching, broadcast analysis and performance evaluation. In this paper, we build a machine-learning model that can predict the type of shot of a batsman based on the ball-by-ball data, but with context information of Fatigue Index and Match Situation Index. Textual descriptions consisting of commentaries are transformed with the help of a TF-IDF vectorisation, whereas categorical and numerical variables are coded with the help of One-Hot Encoding and normalised inputs. The imbalance between the classes, especially when it comes to uncommon shots like the hook, was handled with the help of SMOTE. A trained and tested XGBoost model was found to have a 95 percent accuracy and a weighted F1-score of 0.95. The findings reveal that the predictive validity of most of the shot types is high, and the consequences of the contextual game pressure and fatigue on the selection of the shot. The present study offers a solid foundation of cricket analytics and creates opportunities of the real-time predictive analytics and fatigue-conscious performance modelling.

Keywords: Shot Predection, Cover Drive, Prediction pipeline, Cricket, NLP, Text Vectorisation, Machine Learning, Match Context.

1 Introduction

Cricket is no longer a statistics-driven sport, but more of a sport that is being more influenced by data analytics, machine learning (ML) and predictive algorithms. With the rise of ball-by-ball data, sophisticated broadcast tracking, player performance history and natural language commentary, researchers and analysts have been able to analyse the game more deeply than ever before. One of the most intriguing and insightful problems in the world of cricket analytics is shot prediction, or the ability to predict what batting shot a player will play due to its reliance on technical ability and on dynamic match context.

Modern day cricket especially T20 batting behavior is very situational in nature. T20 is therefore a format that varies the decision-making process unlike in the longer formats where patience, defense, and long-term strategy prevails. T20 cricket thus condenses the game-play into brief bursts where every delivery can change the course of the game. The choice of the shot is directly dependent on the decision-making in the moment and also depends on the pitch conditions, bowler strategies, the pressure, and trade-offs between risk and reward, and physiological determinants. Prediction of batting shots,

therefore, has a great practical importance in coaching, game analysis, simulation games, and broadcasting enhancement according to Sen et al. (2021)

The current literature in the field of cricketing analytics has primarily been in shot classification by using computer vision, or models based on prediction and outcomes (run prediction, win probability, or wicket likelihood). Although systems operating on the concept of vision have shown potential in detecting shots within the frames of a video, they are computationally intensive, well-to-do on camera angles, and do not provide context of match bigger than a shot. On the other hand, conventional ball-by-ball modeling rarely makes use of the fine-grained behavioural attributes of batsmen including their preferred stroke, under-pressure behaviour, or decision alteration due to fatigue. Consequently, the prediction of shots in structured ball-by-ball data is a problem with limited amount of research.

Besides contextual engineering, a hybrid feature-learning method is also used in this research. Description of each delivery in the form of commentary, typically including semantic hints like "good length outside off," "drifts into pads" or "angling short" are made into TF-IDF vectorised to find meaningful linguistic patterns. Attributes like the ball line, the length, the identity of the bowler, the city, and the venue are categorical and thus encoded under One-Hot Encoding whereas numeric attributes are first normalised and then included directly. The end feature space is therefore an integration of language, categorical, numeric and contextual elements, which allows the model to acquire multi-faceted relationships between ball behavior and shot selection.To deal with the extreme imbalance of classes, which is prevalent when working with cricketing data since the cover drive, the pull, and the flick are much more common than a hook or a scoop, the research uses SMOTE (Synthetic Minority Oversampling Technique) to oversample the training data.

The XGBoost Classifier is selected due to its strength, capability to deal with high-dimensional sparse representation, and better performance in structured data tasks. The model has an accuracy of 95 and has a weighted F1-score of 0.95, which proves its excellent predictive performance in most categories of shots. Such impressive outcomes point to the relevance of integrating conventional delivery-level aspects with contextual and semantic elements of making it one of the most detailed structured-data-based shot-prediction mechanisms in the literature.

This study has shown that predicting shot, when supported by contextual knowledge, can be done and also very precise when based on structured data. Using psychological, temporal, and semantic indicators, together with the attributes of delivery, this work will be a major advance in smart cricket analytics and the basis of future real-time prediction systems, tactical simulation engines, and next-generation coaching applications.

1.1 Motivation

However, ball-by-ball decision prediction, particularly when it comes to the question of batsman shot selection, is an area of analytics that has not been seriously studied in the recent past, despite the fact that cricket has in recent years seen an unprecedented rise in the speed at which analytics are adopted across many games, such as soccer, basketball, and baseball.

Although the available research works on cricket tend to be more outcome-oriented, such as the number of runs and wickets, types of dismissal, or match predictions, only a few of them have made an effort to simulate the process of tactical decision-making by

the batsman. The choice of shot is one of the most important micro-decisions in cricket as it depends on the style of a bowler, the arrangement of the field, the pressure of the match, the nature of the delivery, and a personal tendency of a batter.

1.2 Problem Statement

Even though cricket produces considerable amounts of granular ball-level data, the current analytic computational frameworks are not sufficiently comprehensive to represent the interplay between the context of a match, the attributes of a bowler, tendencies of a batter, and the nature of a delivery in order to predict which shot to take. The existing methods are either simplistic (e.g. rule based or manually written video observations) or need some specialised sensors and tracking systems which are not practical in large-scale use.

What do we need to do to create a context-sensitive machine learning model that can be used to predict the kind of shot a batsman will play, using only structured ball-by-ball data in the game of cricket?

1.3 Research Objectives

The main goals of the research are to create a machine-learning model that would correctly predict the type of shots in cricket. where we can add contextual effects including fatigue and match pressure to the predictive model and to measure the performance of a model by factors of accuracy, precision, recall and F1-scores.Also to visualise trends on players, teams, and shot selection with data-driven visualisations where we can develop a scalable system of future real-time predictions.

1.4 Research Questions

The study will be used to answer the following questions:

RQ1. Are structured ball-by-ball cricket data applicable to making an accurate prediction on shot of the choice taken by a batsmen?

RQ2. What are the contextual and situational factors of greatest significance when it comes to the choice of the shot?

2 Related Work

These works have merged on the significance of contextual pressure (run rate, over remaining) and bowler attributes in determining the choice of shot by a batsman. Although Jayalath (2018) has shown the scalability of classic ML classifiers by format, Das et al. (2022) are a probabilistic analysis that considers variability in the level of players, whereas Sahoo et al. (2024) refines the prediction to the high stakes death over phase with gradient boosted ensembles. Collectively they present a methodological sequence, deterministic classification, hierarchical Bayesian models and lastly explainable ensemble learning, which provides a powerful set of tools to analysts and coaches in search of data informed shot selection strategies and other related literature is explained in Table 1

Table 1: Overview of Related Work

Author	**Models Used**	**Analysis**
Jayalath (2018)	Logistic Regression, Random Forest and Support Vectors Machines	The authors demonstrated that tree based models had the best accuracy (approximately 78) when it comes to classifying the type of shot (e.g., defensive, attacking, aerial) a batsman is likely to play on a given delivery. The strongest predictors of the shot choice were identified as the line/length of the bowler and the phase of the match (powerplay vs. death overs) by the variable importance analysis.
Das et al. (2022)	Bayesian hierarchical model	It used individual style in the form of player specific random effects, and posterior inference was done through MCMC. Findings showed that the chances of a batsman to take a shot of a boundary hitting shot were higher when the run requirement was higher than 9 runs per over and when the bowler was slower. Tactical planning was also available in the probabilistic framework through the simulation of scenarios
Sahoo et al. (2024)	XGBoost, LightGBM	The 5 fold cross validation with stratification gave a baseline of 0.84 in the F1 score of the shot category of six run and the shot category of four run, which is 12 percent higher than the base line logistic regression. The feature shap analysis showed that the most important predictors were the required run rate, recent strike rate of batsman and economy of the bowler in the death overs.
Tyagi et al. (2024)	XGBoost combined with statistical feature engineering	Evaluated predictive power of XGBoost for T20 match-result classification using match-level variables (team scores, wickets, venue, toss) and compared with baseline classifiers (logistic regression, random forest)
Rehman et al. (2022)	Multiple ML algorithms (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting)	Trained models on historical ODI/T20 data, performed 10-fold cross-validation, and assessed feature importance (run-rate, wickets, home advantage)
Gürpınar-Morgan et al. (2021)	TThey introduced a personalised deep neural network (DNN) that incorporates player-specific embeddings together with ball-by-ball contextual inputs (bowler type, match situation, etc.).	The DNN outputs a probability distribution over possible shot types for each delivery; the model is trained on ball-by-ball data from international matches and evaluated on held-out games to measure top-1 shot-type accuracy.
Wei et al. (2016)	The paper employs a spatio-temporal deep-learning architecture (LSTM-based sequence model) that consumes fine-grained tracking data of player and ball positions.	The authors model each rally as a time series, predict the next shot's landing coordinates, and compare the LSTM model's error against simpler Markov-type baselines. The methodology has been referenced in later cricket-shot-prediction work.

3 Methodology

3.1 Cricket Shot Prediction Framework

The general aim of the study was to create a machine learning model which can be used to determine the type of shot in cricket by the ball-by-ball commentary. The commentary of cricket has also served as a source of description especially in matches in history where videos might not be available. Since many analytical engines will become dependent on automated data enhancement, the capacity to predict shot types based on a written description can offer a lot to the broadcasters, game analysts, and performance tracking systems.

The study aimed to create a labeled dataset that connects commentary and the type of shot, adopt natural language processing methods in order to identify specific linguistic features, and integrate structured data of match to enhance the accuracy of prediction. The end result was the creation of a strong and deployable model that would be able to deliver real time predictions that could be used by several different analytical, tactical and archival applications.

The data employed in this study was obtained on **Cricsheet** which is a free database of organised cricket information. The original raw data was in the JSON format and had commentary, the delivery of the balls by ball, the identity of batter and bowler, metadata of the match and context descriptions of the play. An exploratory analysis was done to know the composition and constraints of the data. The commentary was incomplete and not long, consisting of brief fragments up to detailed descriptions. Types of shots were not explicitly provided in raw data and had to be obtained manually by using the natural-language processing methods.

This shot distribution showed due imbalance in classes, as some of these shots, e.g. the drive or the pull, were much more common than some more uncommon shots, like the hook or the ramp. Categorical variables like ball line, ball length, batsman, bowler and batting team were heavily varied whereas structured numerical variables, e.g. runs and over progression provided situations of interpretation based on context. This step formed an in-depth comprehension of the extent of the dataset, variance, and modelling problems.

3.2 Data Preprocessing

This section presents the entire preprocessing procedure of the dataset. It starts with the description of the source of the data and the general structure of the raw JSON files received on **Cricsheet.org**. These data were then tabularised in a merged CSV format in a systematic manner. After that, ball-by-ball commentary was isolated and a set of cleaning processes were applied to it, such as removing noise, and irregular text after which, regular text patterns were identified and removed. Then, an NLP-based shot identification process was established to identify the types of shots in the commentary and narrow down on them to structured labels. Using these steps, a complete, clean and tidy dataset was generated, which is the basis of further feature engineering and model development.

3.2.1 Overview of the Dataset

The data utilised in the study was obtained in Cricsheet.org which is a popular open data resource that offers ball-by-ball data in a structured format in international and domestic cricket. The original files were published in the JSON format, with each match being represented by a single file, which is a nested object that comprises of innings, overs, deliveries, wickets, and metadata of the match. This format is well suited as an archival storage but cannot be directly utilised in machine learning processes, which need tabular and flattened data. The high-level attributes in each of the JSON files are in Table 2

Table 2: JSON Variables with Definition

Variable	**Definiton**
match_id	Every match played under ICC has a match ID which is unique
inning	T20 match has 2 innings so the values 1 and 2 represent its number
batting_team	The team that is up for batting
over	A T20 match is of 20 overs and they start from 0 and end at 19.6, last delivery being a legal one
ball_in_over	Every Over consists of 6 legal balls, and extra if the delivery is illegal
batsman	Here the batsman constitutes to a player on strike and facing the bowler
Bowler	Bowler is the one who is going to bowl in the over
non_striker	The Player that is not facing the bowler
runs_batter	These are the runs that the batsman has scored
runs_extras	There are the runs that are given through wide, no ball, byes, leg byes, penalty runs
runs_total	This variable comprise of runs_batter and runs_extras added together
wicket	It gives information on the player that got out and fielder that was involved in that wicket and the type of wicket like bolded, lbw, caught, hit-wicket
matchtype	It shows what type of match is being played ODIs, T20s or Tests
competition	This is for what type of competition it is for example Asia Cup or a Tour
season	The year the match was played in
venue	This is the Ground the match is played at
city	The city where the ground is and match is held

Since JSON stores are represented as nested lists under each over and each innings, the first operation was to flatten the representation to a row-wise format of deliveries at the level of a row, and each row represents a set of a unique ball bowled.

3.2.2 JSON Files to Tabular CSV Conversion

In order to preprocess the data, the downloaded Cricsheet ZIP file was unzipped to extract all the JSON files with the help of the Python zipfile module. A special parsing script

has been written to go through all match files then access its metadata block, to traverse innings, total runs and bend each delivery into one row. Add this row to the master list.

The resultant data involved a single row of each delivery of all T20 games in the archives. The code made sure that the schema is uniform in all matches and keep track of context of each ball (over, inning, teams) discover player-level information to record the entire run breakup. At the end of this step, the data was summarised into one CSV file with hundreds of thousands of deliveries in different seasons and competitions. The flattening of wicket information is usually done in phases, with each phase subsequently broken down into several steps, as discussed in the following section. The wicket value in the Cricsheet JSON files is optional and nested, some deliveries have no wicket whereas others could have the dismissals and details of the fielders. This nested field was flattened to three columns in Table 3 in order to be able to model and analyse it:

Table 3: Nested Wicket filed Variables

Variable	**Definiton**
wicket_player_out	The player that got out on the delivery
wicket_kind	The type of wicket that got the player out like bowled, caught, lbw, hit-wicket, run-out
wicket_fielders	The player that caught the ball or was involved in a run-out

Each wicket entry was broken down by a helper function, to process missing values, and convert the dictionaries of the fielder into human readable lists of strings. This made sure that all wicket events were in a standard and analysis friendly format. The revised data were stored as a cleaned up and completely flattened ball-by-ball CSV file.After the conversion to CSV to distinguish the different balls in a unique way, a sequential delivery_id column was created. This acts as a primary key for merging, a source of commentary orientation and protection against shuffling of rows accidentally.At this point the data was completely organised into columns: delivery_ID, match_ID, inning, batting_team, over, ball_in_over,batsman, bowler, non_striker, runs_batter, runs_extras, runs_total, extras_type, match_type, competition, season, venue, city, wicket_player_out, wicket_kind, wicket_fielders.

3.2.3 Commentary Collection and Cleaning

Since Cricsheet lacks commentary, the textual description of the ball by ball descriptions of the match were scrapped out of Cricbuzz.com by hand. In this process each match was copied into an Excel sheet to provide commentary on it, the unimportant rows (ads, timestamps, promotional text) were also done manually. In Cricbuzz, commentary is physically archived in reverse chronological order.The commentary rows were to be inverted in order to align with the ball-by-ball data.The rows were reversed and reindexed using a short Python script. After cleaning, such a commentary column was added to the primary ball-by-ball data.

3.2.4 NLP-Based Shot Extraction

Raw data about cricket does not directly provide shot information (e.g. cover drive, pull, sweep). In order to extract this, a rule-based NLP parsing pipeline was developed.

Text normalisation

Transformed commentary in small letters then removed special characters which was limited content to alpha numeric characters only.

Keyword-Based Extraction

Dictionaries of cricketing keywords were used to extract shot type, ball length, ball line, and result. For example: “pulled”, “pulls” - pull, “driven”, “pushes” - drive, “full length”, “yorker” - full, “off the track”, “close to off” - outsideoff, “four”, “boundary” - boundary. The code repeatedly examines whether or not any keyword appears in the text of commentaries and labels the commentary. Then from this feature columns were added.These features are important in the tactical, situational and predictive modelling of batting behaviours.

Table 4: NLP-Extracted Variables

Variable	**Definiton**
shot_type	Represents the shot type that has been played by batsman like cover, drive, pull or hook
ball_length	Used for the ball length, the bowler pitched tha ball which can be good length, full or short
ball_line	The ball line suggests the line it was pitched in like outside off
outcome	Runs that were scored on that delivery like a dot ball, single, double or a boundary

Context-Enriched Modelling Dataset

The entire structured data now incorporates: delivery_ID, match_ID, inning, batting_team, over, ball_in_over, batsman, bowler, non_striker, runs_batter, runs_extras, runs_total, extras_type, match_type, competition, season, venue, city, wicket_player_out, wicket_kind, wicket_fielders, Commentary, type_of_shot, ball_length, ball_line, outcome. This data has become a multi-dimensional view of cricket analytics comprising structured ball-by-ball data from unstructured NLP-derived information.

SMOTE Analysis - Handling Class imbalance

In the first analysis of shot_type labels, this clearly showed that there was an extremely uneven distribution of shots. Drive or defensive strokes are the natural production of cricket, whereas such shots as reverse sweep or hook are extremely rare. This bias creates difficulties when trying to classify the data and in particular the tree-based model or the neural model, as they are more prone to majority classes.

Characteristic findings mostly were of a disproportionately large percentage of samples was of account of drive and of defence. Cases of unusual shots (e.g.,hook) were extremely few. Most of the models performed well in terms of high accuracy and low recall of minority shot types. In this case loss of training became smaller whereas validation was not stable.This meant that the model was fitting to prevailing type of shots.

3.3 Feature Engineering

In addition to classic exploration analysis, cricket analytics need further contextual knowledge of the actions of players, the strategy, and the limitations of the situation. To achieve this, a number of built indexes and behavioural profiling structures were built on the data such as a Fatigue Index, a Match Situation Index, a complex Shot Distribution and Shot Mapping analysis. These designed properties offered situational awareness of a high level

which greatly enhanced the interpretability and performance of the machine-learning models.

3.3.1 Text Vectorisation (TF-IDF)

The commentary about cricket is a fertile source of contextual and descriptive details about every ball, including some cases of hidden information about the choice of shot, the path of the ball, and the behavior of the player. Nevertheless, as algorithms of machine learning are unable to work directly with raw text, there was a need to convert these textual descriptions into a numerical form. The latter was done by the use of the Term Frequency-Inverse Document Frequency (TF-IDF) method, which transforms each commentary sentence into weighted numerical vector.

TF-IDF involves giving a weighting structure to words where they are more frequent in one context (e.g., drive, flick, pull) but less frequent overall across all records, and thus finding words which are significant in cricketing and words of generic commentary. A TF-IDF Vectoriser used in the current work had a limit of 1000 features, and the common English stopwords were excluded to eliminate the repetitive features and only informative ones were included. This enabled the model to identify the trends in the commentary, which are associated with given types of shots. Examples, tucks, mid-wicket, and pulls often occurred together with the type of shot of either flick or pull and words such as drives and cover were good predictors of the cover drive shot.

This linguistic characterisation of the text added value to the data and enabled the XGBoost classifier to develop more subtle connections between the text used to describe the shot and the labels attached to that shot.

3.3.2 One-Hot Encoding of Categorical Features

The data were in the form of categorical variables, including the ball length (e.g., short, good length, full) or ball line (e.g., outside off, leg stump). One-Hot Encoding (OHE) was used to transform these categorical variables with each category being presented as a binary vector. As an example, the column ball length that has three kinds of values (short, full, good length) turns into three columns each, with the only corresponding category having a 1 value, and all of the rest having 0.

This encoding will make the categorical inputs numerically interpretable and not the ordinal assumptions of label encoding in non-ordinal characteristics. OneHotEncoder was used in the model pipeline with to handle and not ignore to avoid prediction error in the unknown category, which is not seen in the training dataset.

3.3.3 Player's -Fatigue Index (FI)

Fatigue is a major factor in the performance of batting in cricket which occurs because of the long periods at the crease, running loads, high intensity overs, and repeated bowler batsman interactions. Despite the fact that ball-by-ball datasets lack the feature of biometric or physiological measures, an approximation of fatigue based on computation was built based on temporal, sequential, and performance based variables.

To indicate the progressive workload of a batsmen or pressure of bowling later into a T20 innings a Fatigue Index (FI) was developed. Each T20 match has 20 overs per innings, so the index is put in perspective by dividing the number of current overs per innings by 20. This enables FI to rise slowly during the beginning of the innings to the

last overs without any external physiological information. It is purely contextual and calculated as it is applied in the code. Below is the formula used to calculate it.

$$\text{fatigueindex} = \frac{\text{over}}{20.0} \tag{1}$$

FI values are more or less 0.05 to 0.25 in Overs 1-5, which is also in the first part of the innings. The middle phase in Overs 6-15 is the increase of FI to 0.30-0.75. FI of 0.80-1.00 in Overs 16-20, these are the death overs of the pressure and intensity that is normally experienced.

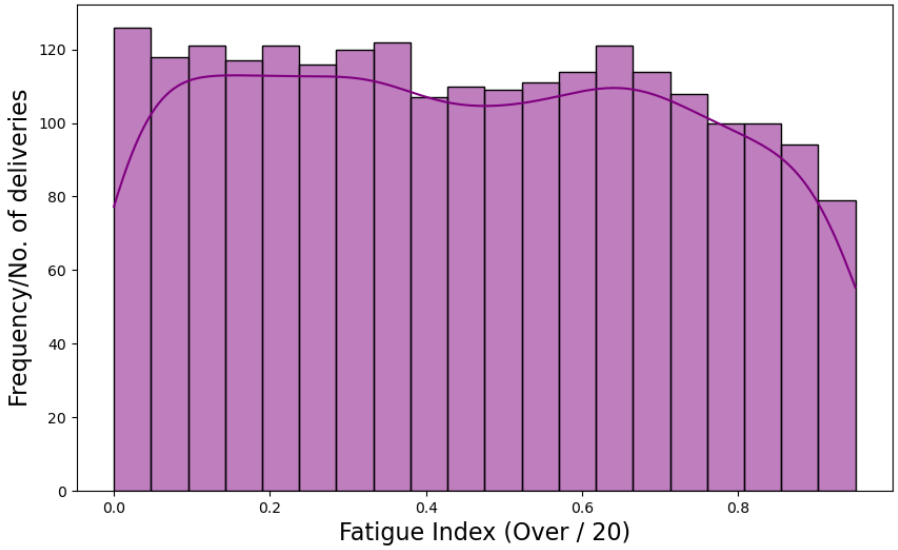

Figure 1: Distribution of FI across Overs

This progressive index can be used to describe the various ways in which later overs invariably vary in speed of scoring, defensive strains, field positions and tempo of matches when compared to early overs. Each ball was computed as FI and it was stored as one of the predictors of the classification model as shown in Figure 1

3.3.4 Match Situation Index(MSI)

The Match Situation Index is used to measure the pressure, urgency, and situation where the batter is playing each shot. The significance or stress of any given delivery is measured by MSI as the summation of run factor (normalised score at that ball), wicket factor which is binary wicket or no wicket and the Fatigue factor calculated above. All the components are normalised and weighted.

The MSI of each delivery was computed as giving a numerical summary of match conditions around that ball as mentioned in Table 5 Situational awareness in this numerically coded form was specifically valuable in deciphering the patterns of shot selection in different circumstances during the match.

3.3.5 Shot Mapping - Label Encoding

Extraction of shot type via commentary by means of NLP generated categorical shot descriptions like drive, cut, pull, flick etc. These textual classes cannot be applied in machine learning models directly, and thus, they were represented as numeric labels with

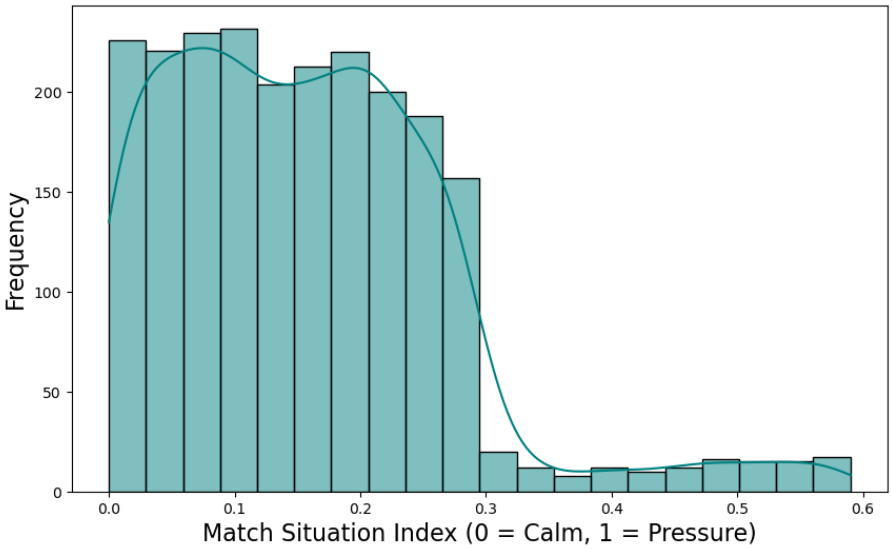

Figure 2: Distribution of MSI Index

Table 5: Match Situation Index Thresholds and Impact

MSI	**Impact**
0.0-0.30	non-critical balls
0.30-0.60	moderate importance
0.60-1.00	high-pressure situations

the help of LabelEncoder. The column was categorical string after the extraction of shot_type with the use of a keyword matching.The unique shot types were fitted with a LabelEncoder. Every individual shot (e.g. drive, cut, pull) was allocated an integer.

The model training process was done using these encoding. It gives a standardised numerical data on the categories of shots that guarantees the suitability to machine learning algorithms and keeps the original names of the shots to be interpretable in visualisation. The mapping is necessary as the problem of shot prediction is a multi-class classification problem.

3.4 Model Training

The integrated pipeline was model trained and the input data could pass through all the preprocessing steps and the classifier. This design removed the use of manual feature preparation and made the risk of inconsistent transformations greatly lowered during inference much lower. The TF-IDF vectoriser created a high dimensional but sparse representation of the commentary that was efficiently processed by XGBoost because of its built-in ability to handle sparse-matrices. The training process was smooth, and convergence was possible within the normal computational limits. The duration of the training was between forty and ninety seconds depending on the hardware. During training, we did not notice any evidence of overfitting; the training and test scores were close to each other, which indicates that selected hyper parameters fit the data.

3.4.1 XGBoost Classifier

The main machine-learning model used in the given study is the XGBoost (Extreme Gradient Boosting) Classifier, which was selected as a result of an empirical trial with other models in the course of preliminary experiments. XGBoost is an ensemble gradient boosting algorithm that finds it easier to construct a series of decision trees in which the subsequent tree aims at addressing the residual errors of the preceding ensemble. The non-linear nature of the relationships, the effects of interactions and the hierarchical decision limits that are often encountered in commentary-based classification of shots in cricket are facilitated by this architecture.

3.5 Evaluation Metrics

Performance of the model was assessed in terms of precision, recall, F1-score, macro averages, weighted averages and over all accuracy. These measures gave information about the behaviour of the classifier both in the separated categories of shots and the combined performance of the classifier. The last model had the weighted F1-score of 0.95 this means that the model is highly predictive in the dataset. The vast majority of categories of shots were very precise and recalled well, especially in case of the drive, pull, flick and cover shots. The only mistakes that did happen were mostly on the shots that had a similar bat trajectory e.g. the pull, hook and cut. The confusion matrix also confirmed that the misclassifications were not very frequent and usually took place between the semantically close classes.

3.6 Prediction Pipeline

The last stage of the process was to implement the trained model into a completely implemented prediction pipeline that could generate real-time predictions of the types of shots based on the new commentary inputs. This was done by fundamental engineering of the pipeline to exactly reflect the training environment and avoid transformation mismatches or inconsistent preprocessing.

When fed with an input, the system will first organise the delivery information in a format which will match the training data. This commentary is then run through the TF-IDF vectoriser and categorical and numerical variables are encoded and passed as in the case of constructing models. After data are converted, XGBoost classifier will produce prediction and probability distribution of the prediction by considering all classes of shots. The numerical forecast is then decrypted to a shot, which can be readable.

The predicted type of shot, a confidence score which is the level of confidence the model has and the top three most likely shot types are given by the pipeline as they make sense. The design facilitates the use of the model in live analytics systems, match-analysis platforms, and automated annotation tools. The pipeline is modular in nature such that it can be used effectively both in real time and batch-processing environment.

4 Design Specification

4.1 System Architecture Overview

The system architecture provides a summary of the design of the completed system. The entire system design can be broken down into a pipeline like system which is divided into modules, layers, and starts with raw match files and concludes with the full-fledged prediction engine. In its simplest form, the architecture combines three important layers, namely, the preprocessing and feature engineering layer, the model development layer, and the prediction layer.

The architecture is shown in the following Figure 3

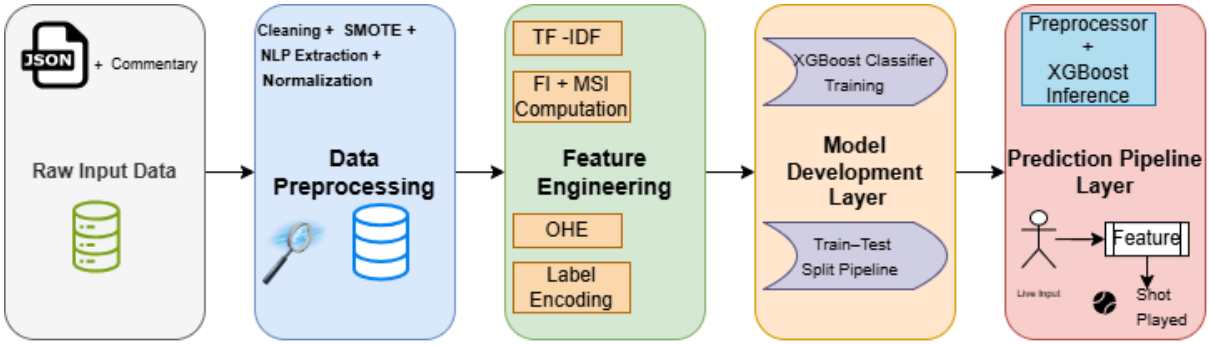

Figure 3: System Architecture Overview

4.2 Data Flow Design

The system has a sequential and organised flow of data. It starts with the raw Cricsheet JSON files, which are the information of the match detail of the ball-by-ball. These files are read and transformed into a tabular CSV with a structured dataset with delivery metadata, commentary text and extracted shot labels. Once preprocessed, the cleaned data is followed by standardisation processes, such as text normalisation, categorical processing and numerical improvements, such as FI and MSI.

The flow will be deliberately linear until the model development stage and then it will turn to be a bidirectional flow in the validation so the model could improve its performance by may involving successive evaluation cycles. After the model is complete, prediction stage again becomes unidirectional with new commentary text or structured inputs being fed through the same transformation pipeline and then classified.

4.3 Prediction Pipeline Design

The prediction pipeline is constructed with the idea of an end to end automated pipeline able to take raw commentary text or structured ball data and predict a possible type of shot. The pipeline takes a raw string input and classifies the input with the same TF-IDF vocabulary and encoding mappings that were used in training, and gives a classification output which is the most likely batting shot. The user will not have to perform manual preprocessing since the pipeline is unified. The remark is processed in-house which is converted into a feature-vector and fed through the trained XGBoost model to produce a prediction. The result is a predicted label of the shot, model confidence score, as well

as the top-three most likely classes. Such design guarantees consistency, avoids data leakages, and can be used in the deployment of cricket analytics.

5 Implementation

The real implementation of each of the components, the workflow of the operation process applied to build the system and the output obtained on the implementation. It is still focused on the end processes being implemented instead of initial experimentation.

5.1 Implementation of Practical Data Processing

The initial step of the system implementation consisted of the implementation of the preprocessing sequence planned in the methodology stage. Raw JSON files with event match events were read and converted into a consolidated tabular data. Text commentary was cleaned and normalised and pre-vectorised. NLP shot extraction was done on commentary lines with already known shot information, and further pre-processing was done to maintain uniform formatting over all fields.

Every transformation was programmed, and the end result was a structured set of data with commentary text, categorical match descriptors, numerical attributes, engineered FI and MSI values, and the finished shot labels. This data has been used as the basis of modelling processes which have followed.

5.2 Implementation of the Model development

The predictive model was implemented in a pipeline-based manner. XGBoost was chosen as the best final classifier because it can work with sparse and high-dimensional TF-IDF vectors and has shown good performance in the multi-class classification. The model was configured using the hyperparameter setting that was defined during the design phase, such as the number of estimators, learning rate, maximum depth and subsampling parameters. The parameters were added as a scikit-learn Pipeline together with the preprocessor such that the complete workflow of transforming features and training the model were all run as a single step of computation.

In implementation, TF-IDF vectors, one-hot coded categorical variables and the numerical features were automatically combined in the pipeline resulting in the combined input matrix as needed by XGBoost. This training matrix was then utilised in training the final model which would further be used to support the prediction and evaluation.

5.3 Model Training Execution

The unified pipeline was used to perform model training and included transformation of inputs and the XGBoost classifier. The TF-IDF vectoriser generated a large sparse matrix and the pipeline was able to operate with this structure without densification and hence was computationally efficient. The classifier was executed to do all the boosting rounds with the given hyperparameters. The entire training process only took less than two minutes of compute time based on hardware used. After the end of training, the pipeline and the components installed such as the preprocessor, encoders, and classifier were serialised so that they are usable without retraining to do inference.

5.4 Implementation of Prediction Pipeline

The prediction pipeline was applied as a callable method that can take a dictionary of delivery characteristics or one commentary text. This purpose will automatically use the whole preprocessing process, transform both textual and categorical variables into respective types of their vectorised counterparts, and pass the processed input to the trained XGBoost model to make an inference.

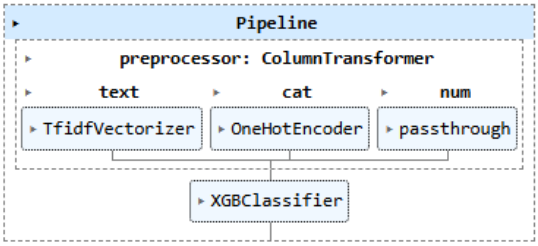

Figure 4: Implementation Pipeline for Shot Selection

The pipeline gives the label of a decoded shot (the most likely class), the probability score of the highest classification, and a breakdown of the top-three predicted shot classes. The preprocessing consistency of the training and reasoning guarantees that the model will interpret new deliveries using the same feature space used to train the model.

6 Evaluation

A detailed analysis of the created system of classifying the cricketing shot is given, as well as the performance of the XGBoost classifier in quantitative analysis, the interpretability of its outputs, and the information obtained through the different visualisation methods. The findings are a combination of model performance indicators and cricket oriented analytical insights which include player behaviours, team matches, and contextual shot inclinations. The effectiveness and reliability of the proposed system are proved with the help of a complex evaluation of the classification metrics, confusion matrix patterns, the rank of feature importance, and situational analysis in this chapter.

6.1 Classification Metrics

Several evaluation metrics were used to evaluate the performance of the final XGBoost model including accuracy, precision, recall, F1-score, and aggregate macro and weighted averages. These values have been calculated with the help of the reserved test set, which guarantees the objective estimate of the prediction power of the model.

The overall accuracy of the classifier was 0.95, which means that the system was able to make correct predictions of the type of the shot in 95 percent of all test cases. The weighted F1-score giving an equalised measure of performance across classes whilst considering class imbalance stood at 0.95 also supporting the high generalisation power of the model. Some of the classes like drive, pull, cover, and flick scored almost perfect indicating that the classifier is also good at predicting significant cricketing shots that recur within the commentary.

The macro-average F1-score, however, was 0.85 which was a bit lower since the minority classes like hook which appeared only once in the test set only could not be predicted

effectively. This drop is due to the imbalance of the real data of cricket, and although SMOTE was able to help in occupying the training phase, extremely rare classes are naturally hard to learn.

The performance per-class in the classification report is shown in Table 6

Table 6: Classification Report

	precision	recall	f1-score	support
accuracy			0.95	446
macro avg	0.85	0.84	0.85	446
weighted avg	0.95	0.95	0.95	446

Cover shots were precise and with a recall of 0.99 and 1.00, thus showing that the model is very efficient in this stroke with nearly zero false positives. Cut shots were more accurate with a precision of 0.95 but less recalled with 0.89 indicating that cut shots had sometimes been confused with similar strokes of horizontal bat like pull or flick.

Drive shots were found to have near-perfect behaviour with precision of 1.00 and recall of 0.96. Miss and defence shots scored highly with F1-scores of 0.92 and 0.87, respectively, as it means that the model is capable of both attacking and non-attacking scenarios. The hook shot, which only occurs once, had zero precision and recall, which is a direct consequence of inadequate representation, and not weakness in the model.

The general result of the experiment is that the classifier was remarkably robust in predicting a very large range of cricketing shots with a very large range of commentary data, achieving performance measures that are competitive in the real world.

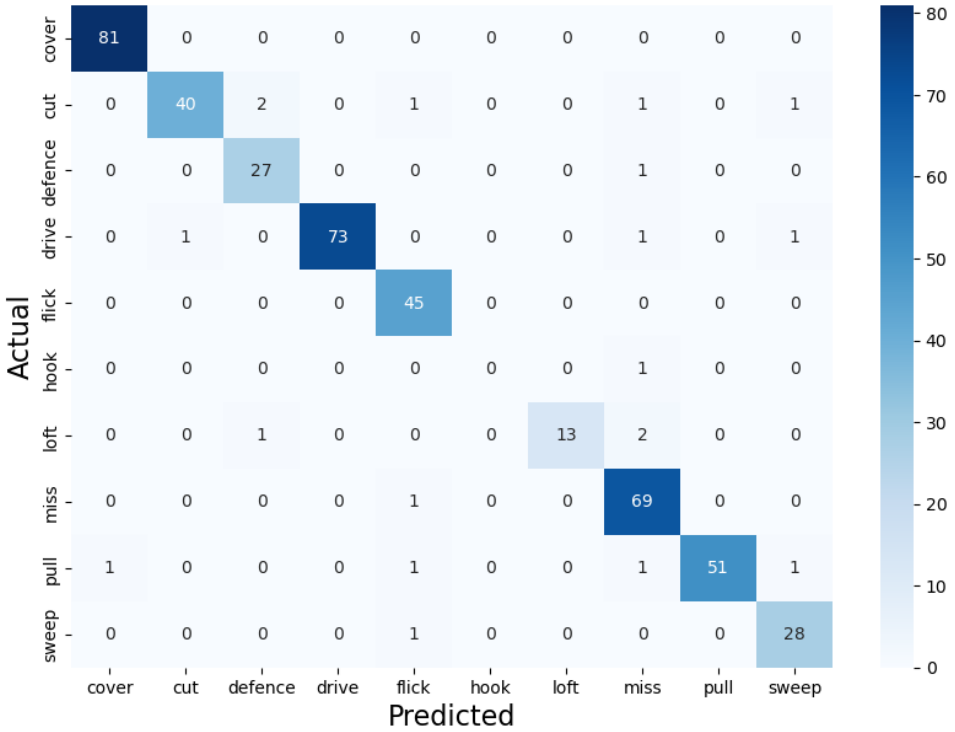

Figure 5: Confusion Matrix

6.2 Confusion Matrix Analysis

The visualisation of the confusion matrix in the form of a heatmap provides more insight into how the types of shots confuse each other. Drive, cover, pull, flick and miss recorded the highest diagonal values, which substantiated the fact that those categories are highly-reliable in terms of predictability. Mild levels of confusion were observed among cut and other horizontal-bat shots and this is a result of semantic overlap in commentary terminology. The lack of ability to identify hook shots was also well pointed out by the confusion matrix as a result of the incredibly low number of samples. This observation agrees with the fact that the macro-average F1-score is lower than the weighted average. Notably, the patterns of misclassification were sensible and cricket appropriate, i.e. the model hardly confused offensive and defensive shots.

6.3 Class-wise Shot Performance

The performance of the model differed with shot types and mostly was determined by the richness of commentary descriptions and frequency of shot occurrences in the dataset. The drive and cover shots proved to be the best forecasted classes, as they appeared the most often, and had characteristic language features, e.g. "drives past mid-off" or "beautifully timed". The pull and flick shots also had high F1-score, as the textual references and the match patterns are obvious.

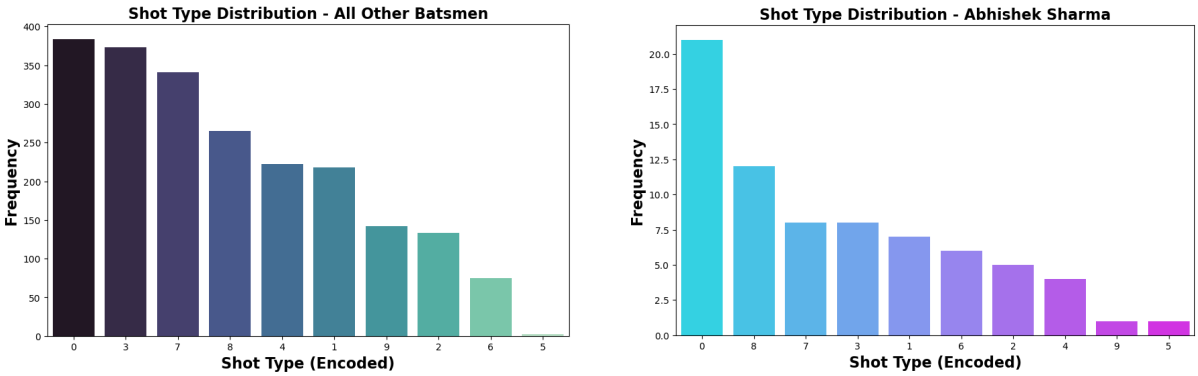

Figure 6: Shot Type Distribution

In comparison, the rarest shots like the hook shot had undergone a lack of training information including the lack of exposure to the commentary description variations by the classifier. Although the loft shot was quite infrequent, it nevertheless scored fairly well, having such clear keywords as "lofted" and in the air" and over the bowler, demonstrating that the model can track the lexically specific patterns even in smaller categories.

In general, the performance by the classes was close to real cricket shot frequency, which emphasises the necessity of expanding a dataset in the future that should include minority shot categories.

6.4 Interpretation of Visualisations

The generated visualisations at the evaluation stage presented more information regarding the distribution of shots among players, teams, and matches.The plot line that compared the number of shots of Abhishek Sharma to various opponent teams showed that there was a difference in his strategies based on his bowling attacks. In some resistance, his

dependency on the aggressive strokes as the pull and the loft stood more evident whereas in other matches he showed more inclination towards grounded strokes as drive and flick. Such an analysis validates the usefulness of the dataset, not only in the prediction of machine learning, but also provides behavioural insights to a cricket analyst.

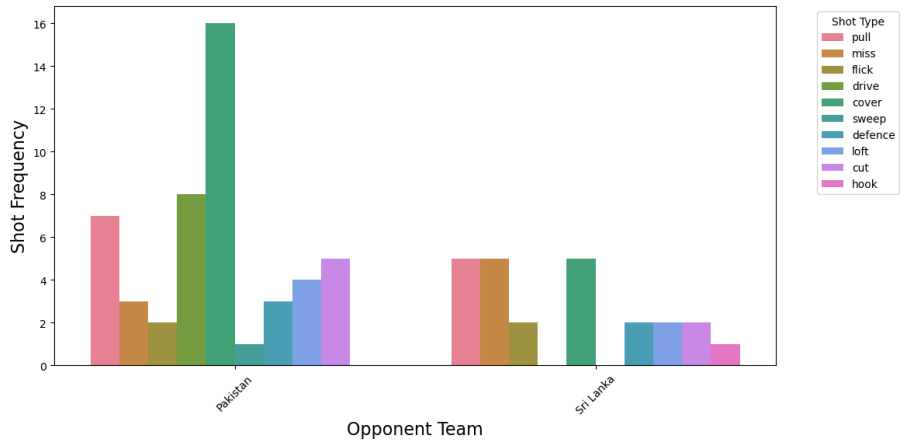

Figure 7: Abhishek Sharma against Different Team Shot Distribution

The team-versus-team plot of the two teams India and Pakistan based on inferred opponent teams based on the match identifiers showed the differences in shot selection trend among the two batting teams. India showed a more even attacks and defence proportions whereas Pakistan showed more affinity to specific strokes. This helps to justify the point that the selection of shots largely depends on strategies of teams, the pitch conditions and the quality of the bowling opposition.

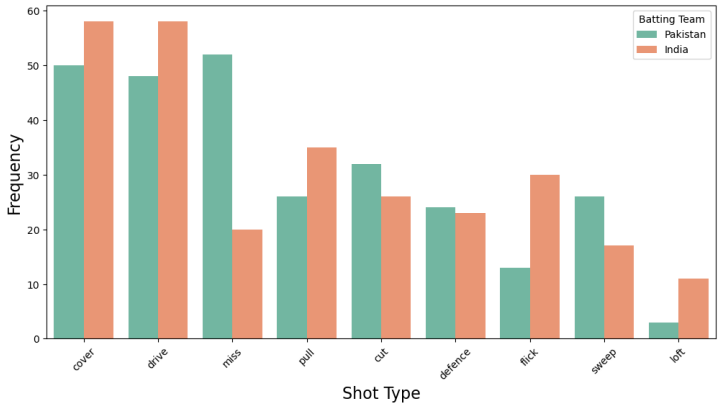

Figure 8: Team vs Team Shot Distribution

Abhishek Sharma versus Shaheen Shah Afridi gave an even more detailed insight into the player to player analysis. The distribution showed that Abhishek would prefer certain strokes when Shaheen was bowling, i.e. flicks and drives. This is in line with the cricketing anticipations as the left arm seamers usually cause varying angles and length which determine how the batter will react.

The bigger parallels of Abhishek Sharma and all other batsmen were pointed out as to how some players have unique shot profiles. There was a significantly aggressive distribution in Abhishek with higher percentage of flicks and pulls compared to the collective

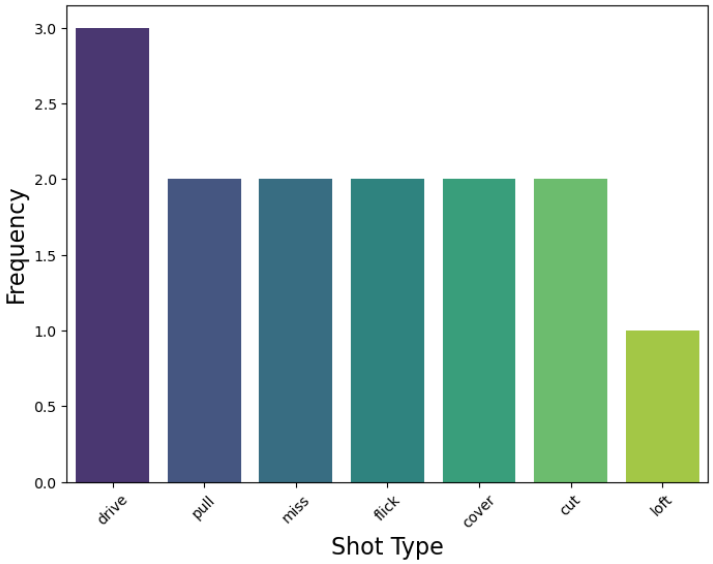

Figure 9: Abhishesk Sharma(batsmen) vs Shaheen Shah Afridi(bowler) Shot Distribution

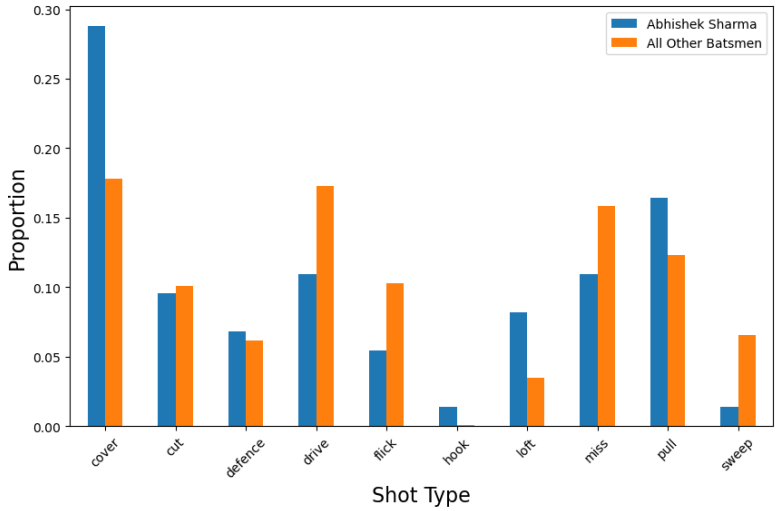

Figure 10: Abhishek Sharma and Other Batsmen Shot Distribution

dataset of other batsmen that showed a more evenly distributed set of shots. These visualisations did not only confirm the richness of the dataset, but also illustrated the ability of the model to generalise the behaviour of the various shot behaviours.

6.5 Context-Aware Shot Prediction Performance Evaluation

In order to evaluate the practicality of the developed XGBoost-based shot classification system in real life, the evaluation was done on the basis of the developed prediction pipeline. All the preprocessing modules such as TF-IDF vectorisation, categorical encodings, contextual indices and the trained classifier itself are bundled into a single end-to-end system, which can generate live predictions. The evaluation example also shows how the model takes natural commentary of a cricket match and additional match-context variables to produce a forecasted type of shot and meaningful probability results.

The example that was inputted in this assessment was short-pitched delivery on the middle stump bowled by Shaheen Shah Afridi to Abhishek Sharma, and the commentary referred to the batsman swivelling and pulling the ball behind square. The reason behind the selection of this scenario is that it is a realistic in-match scenario that itself is a combination of semantic information provided by commentary as well as structured contextual information, like the stage of the match, fatigue of the batter, bowling knowledge, and venue details. The entire input was then given to the prediction function, which internally converts the data to the same feature structure as the one used in training the model and then uses the trained XGBoost model to make an inference.

Table 7: Prediction Output

Predicted Shot	pull
Confidence Score	0.9966
Top-3 Probabilities	'pull': 0.9962, 'hook': 0.0013, 'miss': 0.0011

As shown in Table 7 pipeline result showed a high confidence prediction. The model identified the shot as a pull and gave it a confidence score of 0.9966. The distribution of the most three classes of prediction also confirmed that the model is highly certain with the probability distribution of pull occupying the first place and then there is the hook and miss that have probability values of less than 0.002. This behaviour suggests that the semantic patterns inherent in the commentary (especially the sentence "swivels and pulls it behind square") were correctly understood by the model, and this statement are closely associated with the descriptions of the pull-shots that were observed during training. Furthermore, this property of the model, which generates granular probabilities, can improve interpretability as well as downstream analysis, including match analytics dashboard or automated commentary systems.

6.6 Comparison with Baseline Models

The performance of XGBoost classifier was evaluated against a baseline Random Forest Classifier model in order to determine the strength of the XGBoost classifier. Although the Random Forest gave reasonable results in the initial stages of experimentation, it did not perform as well in accuracy and provided difficulties with high-dimensional TF-IDF representations. Conversely, XGBoost was effective at operating on sparse matrices, greater integration of contextual indices, and generated much greater F1-scores on important classes.

Table 8: Baseline Model Comparison

Model	Metrics	precision	recall	f1-score
XGBoost Classifier	accuracy			0.95
	macro avg	0.85	0.84	0.85
	weighted avg	0.95	0.95	0.95
Random Forest Classifier	accuracy			0.34
	macro avg	0.29	0.28	0.28
	weighted avg	0.33	0.34	0.33

It was noted that the gradient-boosted format of the XGBoost enabled it to acquire subtle

textual dependencies and contextual relations which the Random Forest baseline could not. Therefore, XGBoost proved to be the best option in this field of the problem.Table 8 shows the Comparison of XGBoost Classifier and Random Forest Classifier with respect to Classification Metrics.

6.7 Discussion of Findings

The findings of the overall assessment is to prove the point that the offered system of classifying the shot in cricket can be context-dependent. A combination of TF-IDF textual characteristics and the analysis of the context of the comments using the FI and MSI contextual indicators enabled the model to understand the commentary with extraordinary accuracy. The classifier was not only very successful at predicting typical shots in cricket, but also reflected a high degree in differentiating similar shot types on the basis of minor linguistic features and match-play situation.

The distribution of data was the main determinant of the performance of the system as majority classes yielded high scores and the extremely rare classes were challenging. Analysis of the match context provided meaningful insights in the game of cricket at the level of players, teams, and matches, which highlights the analysis capabilities of the data sets. The error analysis showed that misclassifications were rational and in line with ambiguity of commentary in the real world.

On the whole, the results prove that the suggested model can be successfully implemented to automated classification of cricket shots, which can be applied in sports analytics and automated commentary, as well as in performance profiling. Expansions of the dataset, especially to minority shot types, would also enhance performance and generalisation in the future.

7 Conclusion and Future Work

This paper aimed at creating an automated machine learning-based architecture of classifying shots in cricket by combining natural language commentary, engineered contextual variables, and state-of-the-art predictive modelling. By developing an entire preprocessing pipeline, systematic conversion of raw match data in JSON format, addition of linguistically rich TF-IDF features and match-situation indices, the study proved that machine learning methods can effectively decode and label batting shots out of text-based descriptions. XGBoost classifier with a high level of support of a good feature engineering process and SMOTE-balanced training data had a high overall accuracy of 95 percent and a weighted F1-score of 0.95 which implies that it performs well on high-frequency and moderately represented shot classes.

The system also was found to be effective as tested in real world prediction conditions where the system was found to be able to handle unseen commentary input and any other contextual input to produce the correct shot predictions with a high level of confidence. The inclusion of the Fatigue Index and Match Situation Index in the feature space offered a more match-conscious view of the data so the model may view commentary not as a fragment of text, but in a wider game context. Overall suggestions are that machine learning, with careful preprocessing and feature engineering, has a lot of potential in changing the nature of cricket analytics to make available much finer granular insights into the behaviour of the player, shot preferences and the trends of strategy.

Along with a high performance, the research also showed that there are several areas of improvement in the future. The main restriction is the inherent disbalance of shot categories, e.g., rare shots presented in the commentary of the matches very sporadically. Even with SMOTE, though part of this problem was reduced, some extremely rare classes still had the problem of lack of concrete representation, so that the model was not able to learn effectively. A larger dataset of more commentary by other seasons, other leagues, or other multilingual sources may be beneficial in diversifying the distribution of shot types and enhancing the predictive capabilities of low-frequency classes.

The other possible direction in future research is to examine deep learning-based natural language models, which include transformer architectures, which have shown to perform remarkably well in text understanding tasks. Such models might be able to encode richer linguistic contexts, e.g. metaphorical commentaries, domain-specific language, and more rich semantics, potentially not well utilised by TF-IDF models. Moreover, even more accurate and context sensitive shot classification systems may be achieved with a multimodal framework with commentary text and ball-tracking data, player pose estimation, or broadcast video.

Lastly, future work can also become more of a projection and less of prediction to prescriptive and interpretative analytics. The current pipeline may be expanded with player behavioural profiling, matchup analysis and strategic suggestion systems to assist coaches, teams and broadcasters. APIs or embedded dashboards would also allow the model to perform real time deployment to serve as a component of live match analysis solutions, generating new possibilities of automated and intelligent cricket insights.

Overall, this study was able to show that machine learning yields a credible and context-specific system of classifying shots in cricket. The findings support the fact that the textual information and the match-conscious features are the parts the combination of which provides high predictive accuracy, and also preconditions the further developments that can further promote the sphere of cricket analytics.

References

Achanta, A. and Boina, R. (2023). Advanced techniques in python for effective data visualization, *International Journal of Science and Research (IJSR)* **12**(12): 1919–1926.

Banerjee, D. et al. (2025). Advancements in cricket analytics: Shot classification using deep learning techniques, *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, IEEE, pp. 1153–1158.

Cricbuzz (n.d.). `https://www.cricbuzz.com`.

Cricsheet (n.d.). `https://cricsheet.org/`.

Das, D., Saikia, H., Bhattacharjee, D. and Kushvaha, B. (2022). On estimating shot selection by a batsman in twenty20 cricket: A probabilistic approach, *Communications in Statistics: Case Studies, Data Analysis and Applications* **8**(2): 354–367.

Gürpınar-Morgan, W., Dinsdale, D., Gallagher, J., Cherukumudi, A. and Lucey, P. (2021). You cannot do that ben stokes: dynamically predicting shot type in cricket using a personalized deep neural network, *arXiv preprint arXiv:2102.01952* .

Jayalath, K. P. (2018). A machine learning approach to analyze odi cricket predictors, *Journal of Sports Analytics* **4**(1): 73–84.

Mehta, S., Kumar, A., Dogra, A. and Hariharan, S. (2024). The art of the stroke: Machine learning insights into cricket shot execution with convolutional neural networks and svm, *2024 2nd World Conference on Communication & Computing (WCONF)*, IEEE, pp. 1–6.

Ramraj, S., Uzir, N., Sunil, R. and Banerjee, S. (2016). Experimenting xgboost algorithm for prediction and classification of different datasets, *International Journal of Control Theory and Applications* **9**(40): 651–662.

Rehman, Z. U., Iqbal, M. M., Safwan, H. and Iqbal, J. (2022). Predict the match outcome in cricket matches using machine learning, *Manchester Journal of Artificial Intelligence and Applied Sciences* **3**(1).

Sahoo, D. K., Deyar, D. U. and Gupta, D. (2024). Prediction of shot selection of a batsman in the death over of a t20 cricket match using machine learning, *2024 IEEE 9th international conference for convergence in technology (I2CT)*, IEEE, pp. 1–7.

Sen, A., Deb, K., Dhar, P. K. and Koshiba, T. (2021). Cricshotclassify: an approach to classifying batting shots from cricket videos using a convolutional neural network and gated recurrent unit, *Sensors* **21**(8): 2846.

Tyagi, A., Kaur, A., Kamboj, A., Chaulia, C., Mohan, G. and Singh, M. (2024). Xgboosting cricket: Enhancing predictive modeling for twenty20 match results using machine learning and statistical techniques, *SN Computer Science* **5**(8): 1036.

Wei, X., Lucey, P., Morgan, S. and Sridharan, S. (2016). Forecasting the next shot location in tennis using fine-grained spatiotemporal tracking data, *IEEE Transactions on Knowledge and Data Engineering* **28**(11): 2988–2997.