

**Can Twitter Data Add Value to Historical Data?:
A Comparative Study for T20I Cricket Match Outcome
Prediction Using Machine Learning**

By

S.P.P.M.Sudasinghe

This dissertation is submitted as a partial fulfilment of the B.Sc. in
Statistics with Computer Science

Department of Statistics,
University of Colombo, Sri Lanka.

March, 2023

Abstract

Due to the dynamic nature of the game and the numerous variables that might affect the outcome, predicting the result of a cricket match before it begins can be a very difficult task. Each match leaves thousands of data points, but real-time information cannot be produced utilizing that data. Social media platforms like Twitter have proven their capacity to produce real-time information in many fields. In this study, a machine learning-based method for predicting the result of the match before it starts by considering historical data and Twitter data is proposed. From 2011-01-01 to 2022-10-14, historical data such as batting and bowling statistics, and Tweets related to matches were scraped, and useful features were derived from those data. Sentiment score is one of the variables that needed to be created from tweets. For that, identifying tweets' sentiments was needed, and to achieve that, sentiment analysis was carried out as a part of this study. The fine-tuned RoBERTa-based model outperformed with a 95.3% F1 score the models created using LSTM and VADER. Using those created variables, three datasets were formulated: one with only historical data, another with only Twitter data, and a third with both types of data. Then several machine-learning algorithms (i.e., Logistic Regression, SVM, Naive Bayes, KNN, Random Forest, and XGBoost) were trained and evaluated on these datasets. The best models were produced by the XGBoost classifier for all three datasets, and it was noticed that removing correlated and least important variables improved the models' performances. The findings suggested that features derived from Twitter can be quite useful for making predictions. The models generated using only Twitter data performed better than the models built using historical data, and the models created using both historical and Twitter data exceeded the performances of both of the individual data models with a 73.7% F1 score. This best model performed better than the bookmakers' predictions with a profit for the T20I World Cup 2022 data by accurately predicting 11 matches out of 14, while the bookmakers predicted only 9 matches correctly.

Declaration

This thesis is my original work and has not been submitted previously for a degree at this or any other university/institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

Candidate: S.P.P.M. Sudasinghe

Signature

Date

This is to certify that this dissertation is based on the work carried out by Ms. S.P.P.M. Sudasinghe under my supervision. This dissertation has been prepared according to the format stipulated and is of acceptable standard.

Supervisor: Dr. S.D. Viswakula

Signature

Senior lecturer,

Department of Statistics,

Date

University of Colombo.

Co-Supervisor: Dr. G.P. Lakraj

Signature

Senior lecturer,

Department of Statistics,

Date

University of Colombo.

Coordinator: Dr. H.A.S.G. Dharmarathne

Signature

Senior lecturer,

Department of Statistics,

Date

University of Colombo.

Acknowledgement

I would like to express my heartfelt gratitude to all those who have helped me complete this study.

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Sameera Viswakula, and my co-supervisor, Dr. Pemantha Lakraj, for their support and guidance during the entire process and for spending their valuable time with this project. I feel grateful for the opportunity to have had them as my supervisors. I want to express my gratitude to Dr. Gayan Dharmarathne and Mr. Oshada Senaweera for their support as the coordinators and for the suggestions they gave me to improve this study. My heartfelt appreciation is extended to Dr. Ruvan Weerasinghe (University of Colombo School of Computing) for the helpful suggestions and instructions given at the beginning of the project. I would especially like to express my gratitude to the academic staff of the Department of Statistics, the University of Colombo, and the University of Colombo School of Computing for the support and knowledge they have provided throughout these four years.

I want to sincerely thank my parents, relatives, and friends for their constant encouragement and support throughout this difficult time. I am also thankful to my friends, who helped me during the data collection and annotation phases. I am grateful for their presence in my life.

Finally, I would like to thank everyone who has helped me since the beginning of this process. Your help, encouragement, and advice have been truly priceless.

*With gratitude and appreciation, I dedicate this dissertation
to my beloved parents.*

Table of contents

Abstract	ii
Declaration	iii
Acknowledgement	iv
Table of contents	vi
List of Figures	x
List of Tables	xi
List of Abbreviations	xiii
Chapter 1	1
1.1 Overview	1
1.2 Background	2
1.2.1 The game of cricket.....	2
1.2.2 Social media opinions	3
1.2.3 Why Twitter?	3
1.3 Motivation for the study	3
1.4 Objectives of the study	4
1.5 Significance of the study	4
1.6 Organization of the thesis.....	5
Chapter 2 Literature Review	6
2.1 Predicting the outcome of a cricket match using historical and statistical data....	6
2.2 Predicting the outcome of a cricket match using collective knowledge (social knowledge).....	8
2.3 Predicting the outcome of a cricket match using historical data and collective knowledge	10
2.4 Predicting the match outcome using both historical data and collective knowledge in other sports.....	10
2.5 Sentiment Analysis.....	12

2.6	Conclusion.....	13
Chapter 3 Theory and Methodology		15
3.1	Research Design	15
3.2	Data collection.....	16
3.2.1	Scraping historical data.....	16
3.2.2	Scraping tweets	16
3.3	Data preparation	18
3.3.1	RegEx (Regular Expression).....	18
3.3.2	Data Annotation	18
3.3.3	Sentiment Analysis	19
3.3.4	One-hot encoding.....	26
3.3.5	Data scaling.....	26
3.3.6	Data resampling	27
3.4	Data Visualization and Analysis	27
3.4.1	Multicollinearity	27
3.4.2	Selection of suitable algorithms.....	29
3.5	Model Training.....	30
3.5.1	Machine learning classification algorithms	30
3.5.2	Feature Importance	34
3.5.3	Hyperparameter tuning	35
3.5.4	K-fold cross-validation	35
3.6	Model evaluation.....	36
3.6.1	Classifier performance	36
3.6.2	Partial Dependence Plot (PDP)	38
Chapter 4 Datasets		39
4.1	Historical Dataset	40
4.1.1	ESPNCricinfo's Statsguru	40

4.1.2	Scraping historical data	41
4.1.3	Formulating the initial historical dataset.....	42
4.1.4	Formulating the final historical dataset.....	44
4.2	Twitter dataset	47
4.2.1	Scraping tweets	47
4.2.2	Extracting basic features from tweets	51
4.2.3	Formulating the “Twitter” dataset.....	53
4.3	Historical+Twitter dataset	55
4.4	Betting odds.....	55
Chapter 5 Data Analysis		57
5.1	Data Analysis – Historical dataset	58
5.2	Data Analysis – Twitter dataset	61
5.3	Data Analysis – Historical + Twitter dataset	63
5.4	Conclusion.....	66
Chapter 6 Implementation and Results		67
6.1	Environments and software packages	67
6.2	Data preparation	68
6.2.1	Data annotation	68
6.2.2	Sentiment analysis	68
6.2.3	Preprocessing data	73
6.3	Performance of the final models	73
6.3.1	Historical data model	74
6.3.2	Twitter data model	79
6.3.3	Historical+Twitter model.....	82
6.4	Check the effectiveness of the proposed methodology	89
Chapter 7 Discussion and Conclusion		91
7.1	Discussion	91

7.2	Problems encountered and solutions	92
7.3	Limitations of the study.....	93
7.4	Suggestions for future work	93
7.5	Conclusion.....	94
	Appendices.....	95
	Appendix A : Snapshots of the datasets.....	95
	Appendix B : Sample of Python codes	97
	Appendix C : Results of the factor score method	104
	References	105
	Internet Sources	111

List of Figures

Figure 3-1- Three phases of the research design	15
Figure 3-2- Steps in phases	15
Figure 3-3 -Sentiment analysis approaches	19
Figure 3-4- LSTM architecture (Understanding LSTM Networks, 2015).....	22
Figure 3-5- Transfer learning architecture (Mohsin Abdulazeez, 2021)	25
Figure 3-6- 3-D PCA plot for linearly separable data.....	30
Figure 3-7- Machine learning approaches	31
Figure 3-8- The confusion matrix	36
Figure 3-9- Classification metrics.....	37
Figure 4-1-Men's T20I team rankings as of Oct 14, 2022	39
Figure 4-2- Cricinfo's Statsguru.....	40
Figure 4-3- An output results page of Statsguru	41
Figure 4-4- HTML code of a Statsguru results page	42
Figure 4-5- The code snippet to identify questions	50
Figure 4-6 -The code snippet to classify tweets with multiple hashtags	51
Figure 4-7- The web application.....	52
Figure 4-8- Application section to identify the sentiment of a tweet.....	52
Figure 4-9- Application section to obtain sentiment and values for variables.....	53
Figure 4-10- oddsportal.com.....	56
Figure 5-1- Bar plot for the target variable's distribution.....	57
Figure 5-2 -Correlation-association plot for the Historical dataset.....	58
Figure 5-3- Scree plot for the Historical dataset	59
Figure 5-4- Factor loading for the Historical dataset.....	60
Figure 5-5- PCA plot for the Historical dataset	61
Figure 5-6- -Correlation plot for the Twitter dataset	62
Figure 5-7- PCA plot for the Twitter dataset	62
Figure 5-8- Correlation-association plot for the Historical+Twitter dataset	63
Figure 5-9- Scree plot for the Historical+Twitter dataset	64
Figure 5-10- Factor loadings for the Historical+Twitter dataset	64
Figure 5-11 -PCA plot for the Historical+Twitter dataset	65
Figure 6-1 -Snapshot showing the fine-tuned model uploaded to the model hub	69

Figure 6-2- Histogram of the number of tokens in tweets	71
Figure 6-3- Feature importance plot of Random Forest model for the Historical dataset	75
Figure 6-4- Feature importance plot of XGBoost for the Historical dataset	75
Figure 6-5- Feature importance plot of XGBoost without 'Venue' for the Historical dataset	76
Figure 6-6- Correlation plot after variable reduction for the Historical dataset	78
Figure 6-7- Feature importance plot of Random Forest for the Twitter dataset	80
Figure 6-8- Feature importance plot of XGBoost for the Twitter dataset	80
Figure 6-9- Feature importance plot of SVM for the Historical+Twitter dataset	83
Figure 6-10- Correlation plot after variable reduction for the Historical+Twitter dataset	85
Figure 6-11- Feature importance plot of the final best model	87
Figure 6-12- Partial dependency plots	88

List of Tables

Table 3-1-Options for scraping tweets and their drawbacks.....	16
Table 3-2- Some Snscape features.....	17
Table 3-3- Special characters in regular expressions.....	18
Table 3-4- One hot encoding	26
Table 4-1- Number of matches played.....	39
Table 4-2- ESPNcricinfo extracted features	43
Table 4-3- Finalized historical features	46
Table 4-4 Hashtags and handles list.....	48
Table 4-5- Hashtags that mentioned both teams	49
Table 4-6- Shortened names of each team	49
Table 4-7- Finalized Twitter features	55
Table 6-1- Best hyperparameters of the RoBERTa-based model.....	69
Table 6-2- Metric scores of the RoBERTa-based model for the test set	70
Table 6-3- Metric scores of VADER model for the test set	71
Table 6-4- Best hyperparameters of the LSTM model	72
Table 6-5- Metric scores of the LSTM model for the test set.....	72

Table 6-6 Final results comparison of sentiment analysis	73
Table 6-7-Primary results of the Historical data model	74
Table 6-8-Results of the Historical data model without 'Venue'	76
Table 6-9- Correlated variables under each factor for the Historical dataset	77
Table 6-10- Results of the Historical model after variable reduction.....	78
Table 6-11- Best hyperparameters of the best model for the Historical dataset.....	79
Table 6-12- Primary results for the Twitter data model.....	79
Table 6-13- Results of the Twitter data model without fans' predictions	81
Table 6-14- Best hyperparameters of the best model for the Twitter dataset	81
Table 6-15- Primary results for the Historical+Twitter data model	82
Table 6-16- Results of the Historical+Twitter data model without T2_Mat & T2_FansPred	83
Table 6-17- Correlated variables under each factor for the Historical+Twitter dataset	84
Table 6-18- Results of the Historical+Twitter model after variable reduction.....	85
Table 6-19- Best hyperparameters of the best model for the Historical+Twitter dataset ..	86
Table 6-20- Final results summarization	86
Table 6-21- Data to check the effectiveness.....	89
Table 7-1- Predictive model comparison using F1 scores	91

List of Abbreviations

CSV – Comma Separated Values

GloVe – Global Vectors for Word Representation

ICC – International Cricket Council

KNN – K Nearest Neighbor

LSTM – Long Short-Term Memory

NLP – Natural Language Processing

ODI – One Day International

PCA – Principal Component Analysis

RegEx – Regular expressions

RoBERTa – Robustly Optimized BERT Pretraining Approach

SVM – Support Vector Machine

T20I – Twenty-Twenty International

VADER – Valence Aware Dictionary and sEntiment Reasoner

XGBoost – Extreme Gradient Boosting

Chapter 1

Introduction

This chapter starts by providing an overview of the research and then covers the background of the study. The motivation to conduct this research is then outlined, along with the objectives and significance of the study. The structure of the thesis is presented in the last sub-section.

1.1 Overview

The potential of social media crowds to predict the outcomes of real-world events, from politics and current affairs to entertainment and sporting events, has become increasingly apparent in recent years. The insightful and up-to-date information given by those social networks may be one of the reasons for that. Beyond this, researchers and analysts have demonstrated that, for making predictions with the collaboration of machine learning techniques, social media data can be a valuable addition to historical data. Football, considered the most popular sport in the world, has been the subject of a significant amount of research of this kind. However, when it comes to cricket, a limited number of studies have been conducted to predict the match outcome compared to football. Many studies on cricket have considered historical and instantaneous features only, and a very small number of studies have considered features extracted from social media opinions. When considering the format, most of the studies have focused on One-Day International (ODI) matches. Out of the few studies done on Twenty20 matches, the majority of them have been based on league matches, while only a few have focused on international T20 matches. Despite this, there is a tremendous interest in the game of cricket, especially T20 matches, due to its high-scoring, energetic, and shorter nature, both socially and commercially. To boost viewership and ad revenue, even broadcasting channels invite experts or pundits to their studios to discuss, analyze, and predict the match winners.

This study aims to predict the match winner before the match begins. As far as the author is aware, no previous research has used both historical data and social media opinions to date to predict the outcome of T20 International cricket matches before the game starts. So, this study may be the first to attempt to predict the winner of a T20I match before the game begins using both crowd opinions posted on social media and historical data. The study

considers the Twitter platform as the source of social media crowd opinions. In addition to predicting the match outcome, there are several secondary objectives in this study: to build a model for classifying the pre-match tweets as positive or negative using an efficient sentiment analysis technique, to study whether Twitter-derived features can provide useful information to predict match outcomes, and to check whether there is any potential profit from pre-match betting using the proposed model.

1.2 Background

1.2.1 The game of cricket

With more than 2.5 billion estimated fans, cricket is the second most popular sport in the world behind football, and it is popular primarily in the United Kingdom and certain former British colonies such as India, Pakistan, and Sri Lanka. Most of these cricket-playing countries now have their own franchise-based cricket tournaments. For example, India has the most famous premier league, the Indian Premier League (IPL), Australia has the Big Bash League (BBL), and Sri Lanka has the Lanka Premier League (LPL). These leagues also support the development and popularity of cricket internationally. Of the three main formats, Test, ODI, and T20, almost every cricket league follows the T20 format.

T20 is the shortest format of cricket, with each team getting to bat for 20 overs (120 deliveries) and usually lasting around 3–4 hours if they do not lose all wickets. T20 matches are more popular among spectators because of these factors as well as the dynamic and exciting nature of that format. However, T20 cricket matches do not have much history. The first T20I was played on February 17, 2005, between Australia and New Zealand. According to “ICC Men’s Twenty20 International Playing Conditions,” 2022, between 30 and 15 minutes prior to the planned start time, the toss for the decision of bat first or field first is conducted under the supervision of the ICC match referee. The captain who wins the toss decides whether to field or bat when it is over. Before the toss, each captain must provide a written list of 11 players, along with a maximum of four substitutes, to the match referee. As there are many aspects like this that cannot be controlled when the game is in progress, team management, coaches, and players prepare well in advance of a match. Therefore, if they can get an idea of what’s going to happen in the match and the result even before the toss, it will be a great help in making their decisions.

1.2.2 Social media opinions

The influence of social media has increased substantially over the last several years, and due to this, it has had a profound effect on society. Facebook, Instagram, and Twitter are prominent among these, and by connecting to these networks, individuals can communicate with each other and express their thoughts to a broad audience. Social media has given many individuals more voice. Since social media offers a lot of real-time information that can be utilized to understand public sentiment, trends, and users' behaviors, it becomes a valuable source for making predictions about future events. Real-world applications of this information include predicting and understanding consumer behavior, forecasting election results, and analyzing player and team performance. For example, in 2021, Phil Lynch, head of media at Manchester United Football Club, disclosed that the club uses fan sentiment graphs based on social media content for each player. He said that, when a player's overall negative sentiment is identified, the club works with the player to address it (*Phil Lynch on Manchester United's Media Strategy and "the Ronaldo Effect," 2021*).

1.2.3 Why Twitter?

Twitter has gained popularity since it was founded in March 2006. According to the latest global overview report (DIGITAL 2023: GLOBAL OVERVIEW REPORT, 2023), Twitter has 556 million active users. 61.2% of them use this platform to stay up-to-date with news and current events. Unlike Facebook postings, which can be private or limited to specific viewers, tweets are public by default and available in real-time, and past tweets can also be easily obtained. Even though Twitter doubled message length to 280 characters in 2017, tweets continue to be shorter and more to the point than other social media posts. They are therefore simpler to analyze and evaluate. The most important aspect of Twitter is the use of hashtags to search and categorize specific topics. The sentiment towards a certain subject or group can be sorted using these hashtags. Due to those reasons, many studies have considered Twitter tweets when carrying out sentiment analysis using social media.

1.3 Motivation for the study

Due to its complexity, predicting and obtaining high accuracy rates in cricket matches, particularly in the T20 format, is still one of the most difficult tasks. Every cricket match generates numerous data points, but extracting certain real-time information like players' injuries, sentiment towards the teams, and match-banned players from that historical data

is impossible. And there are so many parties who like to know the ultimate result of the match before it begins. Especially team management and coaches, because they can get an idea of how the match is going, and it helps them formulate their team accordingly and make changes to their strategies. On the other hand, companies and franchises want to know about public sentiment toward each team so they can decide which team to invest in. Cricket has a large pre-match betting industry, and millions of dollars are wagered on cricket matches. Therefore, building an effective model for detecting sentiment in tweets, determining whether Twitter can provide additional information in addition to the information already available through historical data, and developing an accurate model to predict the outcome of a T20 match before it begins will be extremely beneficial to all of these parties.

1.4 Objectives of the study

The study's main objective is to build a suitable model using machine learning to predict the winner of the Twenty20 International cricket matches before the match starts by considering historical data, and features derived from Twitter.

The secondary objectives of the study are as follows:

- To build a more reliable and efficient model in order to identify the sentiment of pre-match tweets, whether the tweet is positive or negative
- To study whether Twitter-derived features can provide useful information on top of historical data to predict the match outcome
- To check whether there is any profit to be gained from pre-match betting using the proposed model

1.5 Significance of the study

To the best of the author's knowledge, this study is the first effort to develop a prediction model for T20I cricket matches before the match utilizing both historical and Twitter data. The majority of the predicting models created in earlier cricket research have focused on historical features, and some studies have considered only Twitter data. However, this study proposes a machine learning-based approach that enhances the accuracy of match outcome prediction by combining social media data with historical data. When running a sentiment

analysis for cricket tweets, almost all the studies have considered either lexicon-based or machine learning-based approaches to label tweets, but this study attempts to provide a more efficient and reliable model for that. On the other hand, only a few earlier studies examined the performance of their top models in real-life situations. So, the model's performance in this study is assessed at the end of the research to see if it performs effectively in real-life scenarios.

1.6 Organization of the thesis

The thesis is divided into seven chapters, including this first chapter, which gives a thorough overview of the study along with research objectives.

The second chapter, Literature Review, reviews the relevant literature on the study's subject as well as its important ideas.

Next, the third chapter explains the research design, methodologies, and background theories.

The datasets are described in the fourth chapter, together with information on how the data were gathered and the features created.

The study's data analysis procedures and findings are presented in the fifth chapter.

The sixth chapter provides the implementation and the final results of the study.

The seventh and last chapter summarizes the study's main findings, identifies its limits, and provides suggestions for further research.

Chapter 2

Literature Review

Studies on machine learning regarding cricket have slightly increased since 2010. Around 35% of those research studies have focused on game outcome prediction, while only 4% of studies have focused on cricket commentary or media (I. Wickramasinghe, 2022). When predicting the outcome of a cricket match, past studies used two main approaches: using historical data and using collective (social) knowledge (Hatharasinghe & Poravi, 2019). In this chapter, the first two sub-sections are reserved for those two approaches. Combining these two approaches could lead to better accuracy. However, it was noticed that building a model by combining these two methods has not been used much in previous studies about cricket. Therefore, the studies done to predict the match outcome in other sports with this combined method were also considered when carrying out this research successfully. Apart from that, the methods that were used to achieve high accuracy rates in sentiment analysis with less time and manpower are described in another section based on previous studies in different fields.

2.1 Predicting the outcome of a cricket match using historical and statistical data

This is the most well-known method of predicting cricket match outcomes among researchers. According to Swartz (2017), one of the main advantages of doing a study on cricket using historical and statistical data is the availability of reliable data. He mentioned in his paper that the Cricinfo website is the primary source of data and that Statsguru, a user-friendly search engine that is hosted by Cricinfo, is a way to access past results immediately. When considering previous studies, almost all the researchers collected data from the Cricinfo website.

Out of the three formats, ODI is the one many earlier studies considered. The objective of Kaluarachchi & Varde (2010) was to predict the chances of victory in the ODI matches and to develop a tool for that prediction. Before using the classification approach directly, they tried other machine learning techniques, association rule mining, and clustering. As they expected, they got significant results for classification. They considered Naïve Bayes, Decision Tree classification using C4.5, Bagging, and Boosting, and they trained these

algorithms using the WEKA. Their results showed that Naïve Bayes is the best classifier for the concerned dataset. They further observed that winning the toss does not significantly affect the match outcome.

Sankaranarayanan, Sattar, & Lakshmanan (2014) used historical data and the instantaneous state of an ongoing match to predict the ODI match outcome before and during a match. The historical features they used are as follows:

- Average runs scored in an inning
- Average number of wickets lost in an inning
- Frequency of being all out
- Average runs conceded in an inning
- Average number of opponent wickets taken in an inning
- Frequency of getting opposition all out

They developed two different models for home runs and non-home runs. They applied ridge regression and attribute bagging algorithms for this study. By comparing their model with the model proposed by Bailey & Clarke (2006), which was used to analyze the sensitivity of betting markets, they observed that their model significantly outperformed Bailey's model for both innings. They found that their accuracy is between 68% and 70%.

When considering the T20 format, Kampakis & Thomas (2015) carried out a study to predict the outcome of the English County twenty-over cricket matches before the commencement of the game. This study used statistical-based historical data. They developed two main models: the first consisted only of team data, and the second consisted of both team and player data. Win percentage, batting average, and bowling average were some of the variables included in the nine team-level features at the start of their study. Out of Logistic Regression with PCA, Naïve Bayes, Random Forest, and Gradient Boosting, they were able to get the best accuracies from the Naïve Bayes classifier for both models. They got 62.4% accuracy for the first model, and they concluded that, from their second model, it is possible to predict the match outcome in almost two-thirds of instances. According to their findings, using PCA before model fitting adds some advantages. Pearson's correlation score worked better than mutual information gain, the chi-square test, and recursive feature elimination in the case of feature selection for them. After studying the accuracies across the years, they observed that their accuracy is consistently above the odds benchmark. External data, such as social media comments, will increase the model's effectiveness, according to their recommendations for future work.

Lamsal & Choudhary (2018) considered the most famous franchise-based premier league of all time, the Indian Premier League (IPL), and trained six machine-learning models. They used those models to predict the outcome of the 2018 IPL matches 15 minutes before the match started, after knowing the toss decision. They got more than 60% accuracy for SVM, Logistic Regression, Random Forest, and Multilayer Perceptron models. The best F1 score was for the Multilayer Perceptron model, 72%.

Singhvi et al. (2022) tried four different approaches to predict the T20 match outcomes: players' past performance statistics, players' ratings, clustering the players performances, and rating players using an ELO-based approach. For this study, they considered international, domestic, and league T20 matches. AdaBoost gave them the best accuracy at 62%.

Recently, to predict the match outcome of the Bangladesh Premier League (BPL) T20 tournament, Pramanik et al. (2022) proposed two different approaches: predicting the match outcome before the match starts based on pre-match data and predicting the match outcome with all the match information, including post-match data. This data set contained 590 records, which were collected 15 minutes before starting the game. They considered 20 features. Sixteen of them were pre-match features. Toss decisions, venue, and ground weather were also included in their dataset. They used both base classifiers as well as ensemble classifiers. For the model with pre-match data, KNN gave the highest F1 score, 66.1%, and Gradient Boost gave the best results for the complete dataset.

2.2 Predicting the outcome of a cricket match using collective knowledge (social knowledge)

This is the other main approach to predicting match results, but only a few studies have followed this approach. Ul Mustafa et al. (2017) evaluated how well machine learning models performed in predicting the outcome of a cricket match before the start of the game using pre-match tweets gathered from Twitter pages of famous cricket websites like cricinfo.com and cricbuzz.com as well as the official Twitter profiles of cricket teams. They created a list of nicknames along with the most popular hashtags for each team to scrape the relevant tweets. Using those tweets, they calculated three features: tweet volume, aggregated fans' sentiment score, and average predicted score. To find the sentiment of a tweet, they selected linguistic features using the TF-IDF score and used it to group the tweets into positive or negative categories. According to their findings, the total volume of

micro-posts is a good indicator of team ranking. In this research, they did not consider home or away team features because most of the matches were away matches for the considered teams. They tried three base classification algorithms: Support Vector Machine, Naive Bayes, and Logistic Regression, and they also carried out a comparison test of the classifiers. They got up to 75% correct predictions for the large-scale data analysis and verified them on CWC15 and IPL2014 data. Both stream and search APIs were used to collect tweets, and tweets for CWC15 and IPL2014 were gathered daily during matches using the Twitter API. The final results showed that SVM performed better than the other two classifiers. Additionally, for the CWC2015 matches, their model was able to beat the bookmakers' predictions with a profit of 67%.

Kannolli et al. (2017) presented a system to analyze tweets to forecast the winning side and the losing side of a cricket match before it began. Mainly, the study consisted of two stages: classifying the collected tweets team-wise and identifying the sentiment concerning each team. Both of these two stages were based on the Naïve Bayes algorithm. They created a lexicon to classify tweets as positive or negative. This system was tested for both offline and online-streaming tweets, and the test results showed that the suggested methodology accurately completed the whole work within a given time. However, their method did not address the negation words.

As in the study of Ul Mustafa et al. (2017), Anbazhagu & Anandan (2021) created three features: the volume of pre-match tweets, sentiment score, and average predicted score. They also used the same techniques to classify the sentiment of tweets. Three grouping calculations, ANFIS, LDA, and MLDA, were used to evaluate performance. The findings suggest that ANFIS has a distinct advantage over MLDA and LDA.

The main objective of Phulare & Deshmukh (2021) was to perform an algorithm to classify tweets as positive or negative more accurately. After extracting the features, they applied supervised machine-learning techniques, Logistic Regression and Random Forest, to complete their objective. Logistic Regression gave them 79% accuracy, and the accuracy of Random Forest was 69%. They mentioned that in the future, others could expand this idea to predict match winners by including a collection of match data from events such as the IPL, T20, and ICC World Cup.

2.3 Predicting the outcome of a cricket match using historical data and collective knowledge

A. N. Wickramasinghe & Yapa (2019) had two main objectives: predicting the match outcome and predicting the man of the match. To achieve the first task, they built three models: one based only on natural parameters, the second one based only on features retrieved from Twitter, and the last based on both natural parameters and tweets. They considered not only pre-match data but also the complete dataset for the matches. They developed a second set of models to investigate how prediction accuracy changed after every ten overs with only tweets. For this whole study, they considered data related to IPL matches. Tweets were extracted using R Studio through the Twitter API. After collecting the tweets, they performed a sentiment analysis. Up to 85% accuracy was obtained from the proposed sentimental model. They used Logistic Regression, SVM, Naïve Bayes, and Random Forest as their classifiers when training the prediction models. Logistic regression was the best classification approach for both tweet-based and natural parameter-based models, and SVM was the best for the combined model. The combined model performed better than the other two individual models. Out of the set of pre-tweets and tweets within the ten overs-based models, the model built using tweets from the last ten overs provided the best accuracy (78%). They were able to predict the man of the match to a certain extent using a social network analysis. Among the studies aimed at predicting cricket matches, only this study was found to use combined approach.

2.4 Predicting the match outcome using both historical data and collective knowledge in other sports

Since only one study could be found that attempted to predict the outcome of a cricket match using both historical data and social knowledge, the studies on other sports that used this combined approach were considered. Kampakis & Adamides (2014) investigated whether features derived from tweets can be used to predict whether an English Premier League game will end in a home team win, an away team win, or a tie before the game begins. For this study, they used three datasets: the Twitter dataset, the historical dataset, and the combined dataset merged from the two initial datasets. Tweets were gathered using the Twitter streaming API. They made a list of hashtags closely related to the respective teams, along with any possible nicknames. When creating the dataset, they didn't consider the tweets that contained hashtags for more than one team. They discovered that some

nicknames related to EPL teams were used by other international sports teams as well, and they overcame that problem by simply filtering out those tweets. A bag of unigrams or bigrams represented the home and away features of the Twitter model, and each club's statistics represented the features of the historical model. Chi-square and the mutual information index were utilized to preprocess the Twitter bag of words. Out of them, bigrams generated using chi-square provided the best results. They used Logistic Regression, Naïve Bayes, SVM, and Random Forest for the prediction and tested all these models using leave-one-out cross-validation. Finally, they were able to prove that it is possible to predict the outcome of English Premier League games with features extracted from Twitter. The Twitter-based model performed better than the historical statistics model, while the combined model outperformed both individual models. For both the Twitter model and the combined model, Random Forest gave the best results. For the historical model, Naïve Bayes gave the best result. However, the study conducted by Schumaker et al. (2016) for match and point spread prediction only using Twitter used another way to deal with the tweets that contained hashtags for more than one team. They labeled tweets with more than one hashtag with the first club hashtag. They believed that when two or more clubs are mentioned in a tweet, Twitter may have wanted to place more focus on the first club mentioned.

Godin et al. (2015) also wanted to predict the winner of soccer games in the English Premier League (EPL) 2013-14 before the match began, utilizing both statistical data and collective wisdom that was extracted from the tweets. By using the volume of tweets scraped, sentiment analysis, user score prediction, and statistical data, they created four potential approaches to achieving their objective. Then, they tried different ways by combining statistical and Twitter-based features to build the best model. The top 10,000 unigrams were used to create a sentiment classifier for this study using SVM. They trained this classifier on 3,400 manually annotated tweets. Then, using this classifier, they classified the filtered tweets out of the 50 million scraped tweets posted 24 hours before the game into three categories: positive, negative, and neutral. Naive Bayes, Logistic Regression, and SVM were applied to this dataset. They compared their predictions with three different baselines: a naive baseline, which always gave the home team the advantage; predictions from an expert; and the bookmakers' thoughts. Finally, they came to the conclusion that a mix of statistical and Twitter-based information could outperform all three baselines, including expert and bookmaker predictions.

2.5 Sentiment Analysis

Almost every study that used tweets to predict the match outcome used a rule-based or machine-learning approach to find the sentiment of tweets. In order to find a fresh approach to this, it was necessary to take into account different study domains. Lay et al. (2019) noticed that in most of the studies related to sentiment analysis, researchers use manual data labeling, which takes a lot of time and money. Therefore, they suggested a semi-supervised learning strategy using mostly unlabeled data and only a small portion of labeled data from the IMDB dataset. They employed deep neural network architecture on this dataset, and the outcomes demonstrated that the unlabeled data did assist in model training without having a negative effect on the model's performance.

Al-Shabi (2020) focused on five of the most significant lexicons: VADER, SentiWordNet, SentiStrength, Liu and Hu opinion lexicon, and AFINN-111. They made an effort to assess the most significant lexicon utilized in the field of sentiment analysis using data from Twitter. Two datasets, Stanford and Sandars, were used for this study without any pre-processing steps. Their findings indicated that VADER has the highest accuracy for classifying tweets as positive or negative.

Chiny et al. (2021) also discovered that labor and other costs are significant issues in sentiment analysis tasks. As a solution, they introduced a hybrid approach based on TF-IDF, VADER, and LSTM with GloVe embedding. In order to combine these three models, the evaluation scores generated by these three were used as the input to the binary classification model. They used five classification algorithms: Logistic Regression, SVM, Naïve Bayes, KNN, and Random Forest, and applied these models to the IMDB movie review dataset and the Twitter US Airlines sentiment dataset. According to the results, the hybrid model performed better than the three individual input models: TF-IDF, VADER, and LSTM. The results showed that the knowledge gained from the IMDB dataset transferred well to the Twitter US Airlines sentiment dataset.

Ghasiya & Okamura (2021) classified headlines using the cutting-edge RoBERTa sentiment classification algorithm. They achieved 90% validation accuracy with that model, which outperformed the other traditional models. However, neither the original versions of BERT nor RoBERTa are up to date with the current events. For example, those two original models didn't have enough knowledge about COVID-19. To address this problem, Loureiro et al. (2022) introduced a set of language models trained on Twitter over

different time periods. The Twitter-RoBERTa-base-Sentiment-Latest model was based on that research and fine-tuned for sentiment analysis with the TweetEval benchmark after being trained on roughly 124 million tweets between January 2018 and December 2021.

2.6 Conclusion

Due to the inability to find a study that had been conducted to predict the T20 international match outcome before starting the match using historical data and features taken from the crowds' opinions, the previous studies covered in this chapter were used to get an idea for this research. After a proper study, some of the following features used in prior studies were considered when creating the attributes:

Historical features:

- Average runs scored
- Average number of wickets lost
- Average runs conceded
- Average number of opponent wickets taken
- Win percentage
- Venue

Twitter-derived features:

- Tweets volume
- Aggregated fans' sentiment score
- Average predicted score

To find the fans' sentiment score, conducting sentiment analysis on tweets was needed. According to past studies related to the area following are the methods they used for sentiment analysis.

- Rule-based sentiment analysis with a manually created lexicon
- Machine Learning sentiment analysis

However, these methods had some limitations. Therefore, a new method wanted to be considered for this study. The following methods can be considered for that:

- VADER
- LSTM
- RoBERTa

The following machine learning classifiers were selected to train the models:

- Logistic Regression
- Support Vector Machine
- K Nearest Neighbor
- Naïve Bayes
- Random Forest
- Boosting

Chapter 3

Theory and Methodology

This chapter outlines the procedures and methods used to conduct the study. The first subsection of the chapter describes the research design in detail. The next subsections thoroughly describe the theoretical and methodological background of each step of the research design, including data collection, data preparation, data analysis, model training, and evaluation.

3.1 Research Design

In order to develop prediction models, this study was carried out in three phases. Only historical and basic statistical data were taken into account in the first phase, while features generated from tweets were taken into account in the second. The final phase was building combined models with both historical and Twitter-derived attributes.

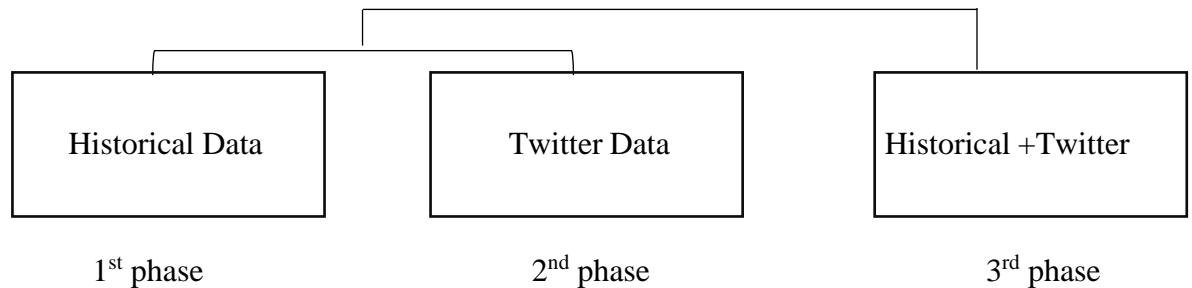


Figure 3-1- Three phases of the research design

Each of these phases consisted mainly of the following steps:

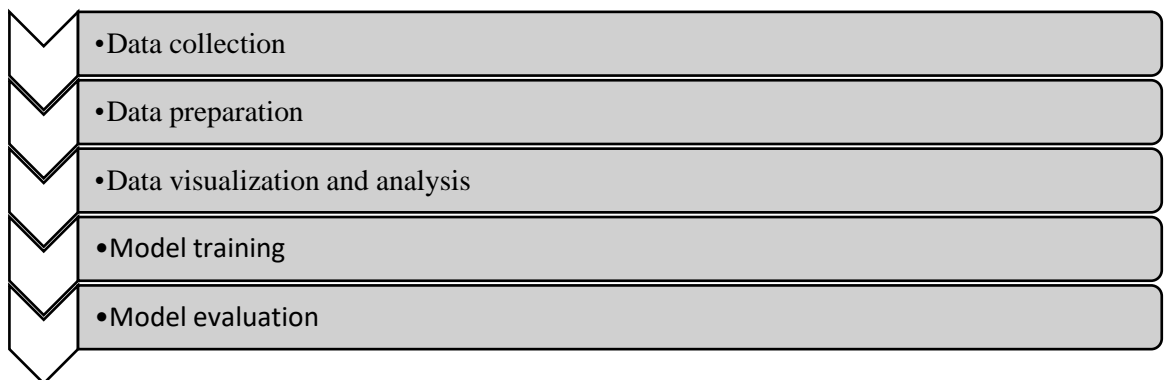


Figure 3-2- Steps in phases

Methods of data collection and preparation are provided in the next two subsections. A detailed explanation of formulating the datasets will be given in the next chapter. After the data collection and preparation steps, the data visualization and analysis step was done primarily to have a good understanding of the datasets. The selected set of models was then trained on the training set, and the testing set was used to assess how well they performed. The model with the best performance was chosen as the best model by comparing those models.

3.2 Data collection

3.2.1 Scraping historical data

The process of obtaining large amounts of data from the web is referred to as "web scraping." "Beautiful Soup Library" is the library that comes to everyone's mind when talking about web scraping. Instead of Beautiful Soup, Pandas Library can easily accomplish this task if the data is available on an URL in tabular format. The `read_html()` function reads data from web pages that contain HTML tables and returns a list of data frames for each table found. However, understanding the structure of the web page is essential in order to scrape the relevant data. A web page's foundation is made up of HTML tags, and the structure of a web page can be specified using these tags. The tag `<table>` is used to generate tables that display data in rows and columns. The `<tr>` tag is used to define rows, and the `<td>` tag is used for existing columns.

3.2.2 Scraping tweets

When considering scraping tweets, there are so many options, but most of them have drawbacks.

Table 3-1-Options for scraping tweets and their drawbacks

Option	Drawbacks
Tweepy	Tweepy only permits historical tweet data collection for up to one week. However, Academic Research Access (https://developer.twitter.com/en/products/twitter-api/academic-research) provides access to the full-archive search tweets, but the problem here is that the applicant has to meet all the requirements they ask for. Tweepy does not allow Academic Research Access to undergraduates. Its application form makes

	it very clear that undergraduates should choose Elevated Access, which does not offer a full-archive search.
GetOldTweets3	This package is no longer available.
Twint	Twint is slower than Tweepy, and there are also some installation-related difficulties.
Octoparse	It requires a lot of time and a paid software.

After considering all of the above options, a Python library called Snsrape (<https://github.com/JustAnotherArchivist/snsrape>) was considered. Snsrape, short for "Social Network Scrape," is a tool that can be used to scrape data from Twitter and other social networking sites. Snsrape does not use the Twitter API. Therefore, there is no need for a Twitter developer account. There is no cap on the number of tweets, and it allows anyone to retrieve all the past tweets from the very first day of Twitter without any limit. Therefore, this is the best way to extract unlimited tweets for a specific time range (Ağrali & Aydin, 2021). But Snsrape also has some disadvantages due to the absence of certain functionalities. However, it is sufficient to scrape tweets. The following are just a few of the many useful features that Snsrape provides:

Table 3-2- Some Snsrape features

Attribute	Description
date	Tweet posted date
content	Content of tweet
replyCount	Number of replies
retweetCount	Number of retweets
likeCount	Number of likes
lang	Machine-generated or assumed tweet language
media	Media object

Snsrape can be used in two ways: via the command prompt, terminal, or a Python wrapper. The Python wrapper was used in this study due to its simplicity. Tweets are scraped according to a given text search query. That query can include hashtags, user handles,

languages, time frames, and other information, and appropriate tweets are extracted in accordance with those inputs.

3.3 Data preparation

3.3.1 RegEx (Regular Expression)

Regular Expressions, commonly known as RegEx, are extensively used in a range of Natural Language Processing (NLP) activities. When there is a pattern to search for in a text, regular expressions are especially helpful (Jurafsky & Martin, 2023). Those patterns come up with a combination of characters and are used to find, replace, and validate the text. The following are some frequently used characters and their meanings:

Table 3-3- Special characters in regular expressions

Character	Meaning
. (dot)	any character besides a newline character
^ (caret)	the commencement of a line
\$ (dollar)	the line's ending
* (asterisk)	zero or more repeats of the prior character.
+ (plus)	one or more repeats of the prior character.
? (question mark)	zero or one repeat of the prior character
[] (brackets)	any character contained in a square bracket.
(pipe)	OR operator, any of the options separated by the operator

3.3.2 Data Annotation

In real-world scenarios, labels are not always provided for each and every set of data. One of the key phases in developing machine learning models for NLP applications is "data annotation," which is the act of assigning labels to certain pieces of data. Out of different kinds of data annotation problems such as text annotation, image annotation, and audio annotation, text annotation is the most widely used in applications. One of them is the analysis of sentiment. Annotating data can be done automatically, semi-automatically, or manually (Al-Laith et al., 2021). Unsupervised learning techniques are used to automatically annotate data, while machine learning methods are used to aid human annotators in a semi-automatic manner. Manual data labeling is done completely by human annotators. Each piece of information is examined before being given the appropriate tag.

However, manual data labeling mainly depends on the human annotator who performs the task. As a result, it could be biased and subjective. To get around these issues, more than one annotator might be asked to annotate the same dataset. Following that, labels may be assigned based on the vote of the majority.

3.3.3 Sentiment Analysis

In this work, one of the Twitter-derived features was built using sentiment analysis.

Sentiment analysis, commonly referred to as "opinion mining," is a subfield of Natural Language Processing (NLP) that uses computational techniques to identify the sentiment in text and classify it as positive, negative, or neutral. Positive sentiment is the expression of joy, pleasure, or acceptance in the text. Negative sentiment refers to the expression of anger, regret, or disappointment. The lack of emotional expression or the expression of a neutral or balanced emotion in the text is known as neutral sentiment. In some circumstances, neutral sentiment is not as important as positive or negative sentiment because the objective is to determine the overall positive or negative reaction to a specific event or outcome. However, these three categories are helpful in determining how the general public and customers feel about certain commodities, facilities, and events.

There are primarily three approaches to performing sentiment analysis:

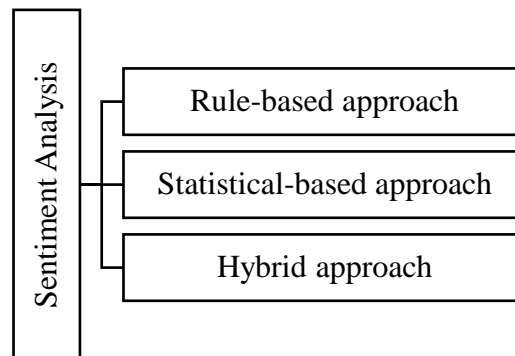


Figure 3-3 -Sentiment analysis approaches

Rule-based approach

The rule-based approach to sentiment analysis, which is a simple but effective method for determining the sentiment expressed in text, uses a collection of established rules and dictionaries. This method determines the mood of a text by identifying words or phrases

that express emotion, such as positive and negative keywords. The overall sentiment of a text is calculated by counting the number of positive and negative words in the text. With this technique, the sentiment of a large text can be easily and quickly determined.

VADER (Valence Aware Dictionary and sEntiment Reasoner) is one of the rule-based sentiment analysis tools and is well suited for short and informal text data in social media, like tweets (Chiny et al., 2021). It represents the sentiment of a text using a combination of dictionaries, sentiment-specific words, and context-based sentiment orientation. For that, it produces four scores, and they are as follows:

- Negative Score: a value between 0 and 1 that quantifies the negative sentiment represented in a text, where 1 denotes the maximum negative sentiment and 0 the absence of negative sentiment.
- Neutral Score: a value between 0 and 1 that quantifies the neutral sentiment represented in a text, where 1 denotes the maximum neutral sentiment and 0 the absence of neutral sentiment.
- Positive Score: a value between 0 and 1 that quantifies the positive sentiment represented in a text, where 1 denotes the maximum positive sentiment and 0 the absence of positive sentiment.
- Compound Score: a normalized value between -1 and 1 that represents the overall sentiment in a text, where -1 denotes the maximum negative sentiment, 1 the maximum positive sentiment, and 0 the neutral sentiment.

The most significant characteristic of VADER is its sensitivity to capitalization and punctuation (Hutto & Gilbert, 2014), as well as its capacity to handle emoticons, slang, and other social media features, which are difficult to handle by traditional sentiment analysis models. VADER is a pre-trained tool, usually, there is no training phase here (Hutto & Gilbert, 2014). To evaluate the performance of VADER sentiment analysis, the ground truth labels are generally compared with their predicted labels (Chiny et al., 2021).

However, the rule-based approach frequently has drawbacks. The dictionaries and rules might only be specific to some domains and cannot adapt to updated phrases, variations in sentiment, or emerging trends. It frequently misses complex negations and struggles to recognize irony and sarcasm in text.

Statistical-based approach

The statistical-based approach utilizes machine learning and deep learning algorithms for accurate sentiment recognition. Machine learning algorithms for sentiment analysis can be further divided into two types, supervised machine learning algorithms and unsupervised machine learning algorithms. Support Vector Machine, Naive Bayes, and Logistic Regression are a few prominent supervised machine-learning algorithms, and the most prevalent unsupervised learning strategy used in sentiment analysis is clustering (Chiny et al., 2021). Supervised machine learning algorithms require labels to train the model, while unsupervised learning approaches do not require labels. Unsupervised learning may struggle to accurately identify sentiment because of this lack of guidance. Therefore, they could not be as credible as supervised learning algorithms. Due to their ability to automatically obtain useful features from unprocessed texts, deep learning models have the potential to be more effective than conventional machine learning models in sentiment analysis (Liang et al., 2017). In recent years, deep learning algorithms like Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) have become increasingly prominent for sentiment analysis tasks.

LSTM also known as Long Short-Term Memory, an updated version of RNN, is widely utilized when building models from scratch for NLP applications like sentiment analysis, and it overcomes the vanishing gradient problem in RNN (Hochreiter & Schmidhuber, 1997). The mood of a text may be influenced by words or phrases that come earlier in the text. In situations like this, LSTM frequently works well. The following snapshot describes the LSTM architecture:

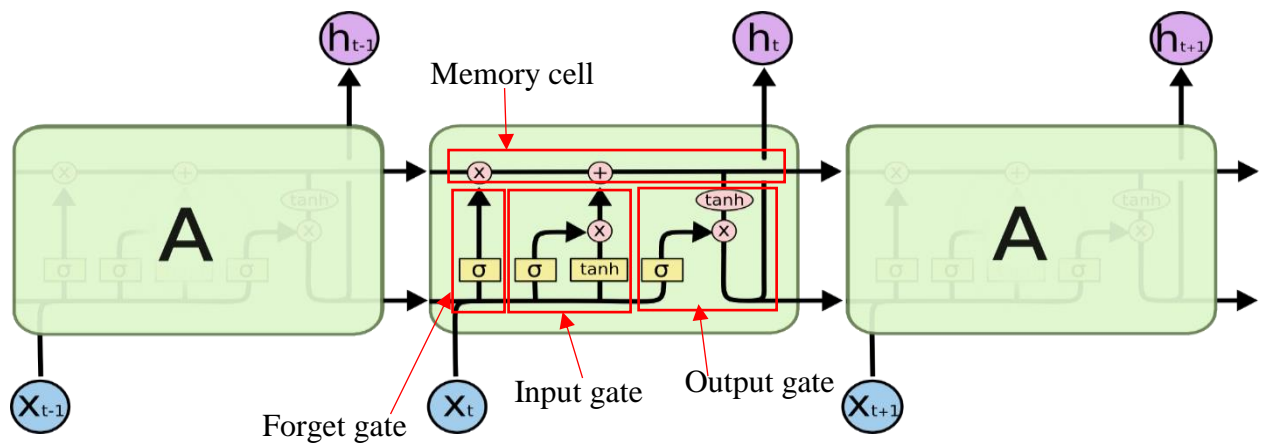


Figure 3-4- LSTM architecture (Understanding LSTM Networks, 2015)

- Memory cell: manages the information flow across each cell by employing Forget Gate, Input Gate, and Output Gate
- Forget Gate: in order to optimize the network, determines which information should be deleted after taking into account its significance to the LSTM. The output of the previous cell (h_{t-1}) and the input at the current time step (x_t) are the two inputs to this gate, and the weight matrices are multiplied by these two inputs. The values that should be rejected are then determined using the sigmoid function.
- Input Gate: adds information to the state of the memory cell.
- Output Gate: chooses usable information based on the current cell state and displays it as an output.

To get the best results from LSTM before modeling, the text should go through the proper pre-processing stages (Chiny et al., 2021). Cleaning, tokenization, lemmatization, and word embedding are the primary components of the data preparation process for LSTM.

Step 1 : Cleaning

The removal of hashtags, user handles, multiple whitespaces, URLs, punctuation, and lowercase comes under the cleaning part.

Original tweet: @YUVSTRONG12 Good Luck Boyssss! <http://t.co/VKEvYUffdM>
#IndvsAus

After cleaning: good luck boyssss

Step 2 : Tokenization

Tokenization is the process of dividing long text strings into a number of tokens, each of which stands for a different word or symbol. Tokens are more convenient to utilize than the entire text.

After tokenizing: ['good', 'luck', 'boyssss']

Step 3 : Lemmatization

Lemmatization converts words into more common representations by reducing them to their root form.

After lemmatizing: ['good', 'luck', 'boy']

Step 4 : Embedding

Words are represented as numerical vectors using a process called word embedding, and the model uses these vectors as inputs. The two types of word integrations that are most frequently used are Word2Vec and GloVe. Out of these two, GloVe, created by Stanford University researchers, captures global contexts better (Pennington et al., 2014). Depending on the problem, GloVe may either be pre-trained or trained from scratch. Pre-trained GloVe embeddings are considered a good place to begin. The word vectors produced by the GloVe model can include 50, 100, 200, or 300 elements. A higher degree of dimensionality can lead to improved performance, but on the other hand, it is computationally infeasible and may cause overfitting. GloVe 50d generates 50-dimensional vectors to represent words, and it is a popular choice for NLP tasks, notably in sentiment analysis, because of its ability to provide a balance between computational efficiency and the ability to capture the associations between words.

When the training dataset gets bigger, statistical-based methods like LSTM often become more accurate. However, when the data labels are unavailable, annotating the huge dataset is not an easy task. Due to this reason, this approach is both costlier and more time-consuming.

Transfer learning-based approach

As mentioned in Chapter 2, most of the previous studies conducted for predicting the match outcome using tweets used rule-based or machine learning-based approaches in sentiment analysis. In order to introduce a new, efficient, and more accurate model, transfer learning-

based sentiment analysis was considered in this study. Transfer learning is the practice of using a model that was originally trained on a large dataset for a particular task as a starting point to address a different but related issue using a very small quantity of labeled data. It is particularly effective in situations where labeled data is expensive to collect.

BERT (Bidirectional Encoder Representations from Transformers), a pre-trained transformer model introduced by Devlin et al. (2018), can be used to perform sentiment analysis on a new dataset using transfer learning. Transformers are a good choice for this job because they are capable of modeling context and making relationships between words in a sequence, which is crucial for figuring out the sentence's overall mood. By using these pre-trained models, developers and researchers can save a significant amount of time and resources that would otherwise be required to train their own models from scratch. Hugging Face, a platform that gives developers and researchers access to state-of-the-art NLP models, offers access to a vast set of pre-trained NLP models, including models, for sentiment analysis. These models can be accessed through the Transformers library in Python.

The advantages of using pre-trained models through Hugging Face can be listed as follows:

- Models on Hugging Face often achieve high levels of accuracy and reliability.
- Hugging Face comes up with a user-friendly interface to make using the pre-trained models and fine-tuning them easy.
- Users can deploy their models easily on different platforms, including the cloud.
- Hugging Face supports environments like PyTorch and TensorFlow. Therefore, it is easy to use pre-trained models in different environments.
- Hugging Face has a very active community that shares their experiences and provides support.

The models offered by Hugging Face were trained on a particular task and a collection of data. Therefore, they may not perform well when given inputs that differ from the training data. By considering this, it is good to assess the models' effectiveness and fine-tune them before applying them in practice.

For this study, the "twitter-roberta-base-sentiment-latest" model was considered, which is available in Hugging Faces's model hub. It is based on RoBERTa (Robustly Optimized BERT Pretraining Approach), proposed by Liu et al. (2019), which is a variation of BERT.

There are so many improvements in the RoBERTa models compared to the BERT. Employing more training data, eliminating next-predict loss, and expanding the batch size are some of them (Zhao et al., 2021). The model considered in this study was trained to predict the sentiment of a tweet with three possible labels: positive, negative, or neutral. This model can be fine-tuned for another new task with binary labels, positive and negative. In a situation like that, as shown in Figure 3-5, the first and middle layers of the pre-trained model are not changed during transfer learning because the model has already been trained to classify tweets. Only the last layer, also known as the model head, needs to be changed by giving a predetermined number of output classes.

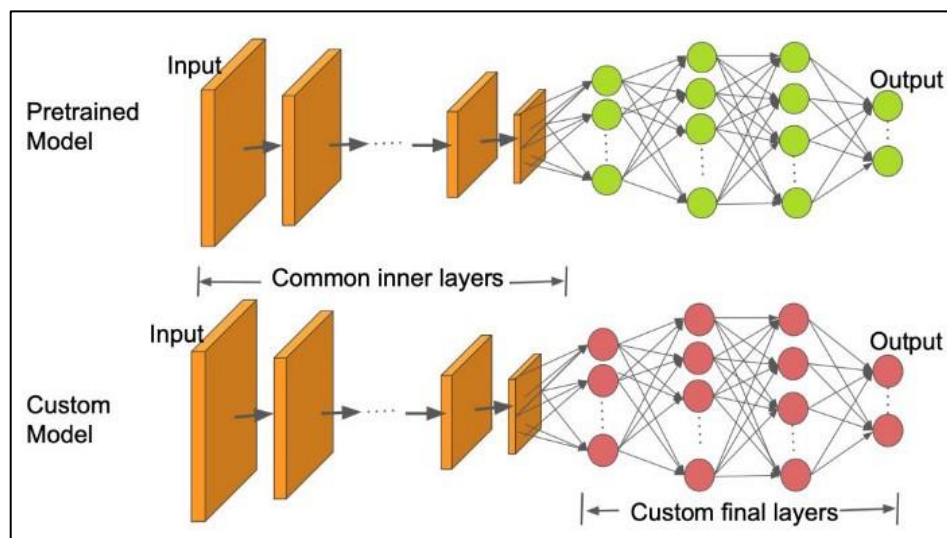


Figure 3-5- Transfer learning architecture (Mohsin Abdulazeez, 2021)

When training the model on a new dataset, training arguments also play a crucial role. These arguments list the numerous hyperparameters, such as batch size, learning rate, optimizer, and the number of epochs, that regulate the training process. Faster training is often accomplished with larger batch sizes, but more space is needed. Convergence could be delayed with a lower learning rate, but the model may be more accurate. More accuracy can be achieved with more training epochs, but there is a possibility that the model may become overfit. Early stopping can be used to prevent that (Cheang et al., 2020). During training, the model's parameters are adjusted using optimizers like SGD and Adam.

3.3.4 One-hot encoding

Not all machine learning techniques can directly interact with categorical data labels. In certain circumstances, the variables have to be transformed into numbers before being introduced into machine learning algorithms. One-hot encoding, which is among the most well-known encoding techniques, is frequently used to transform nominal data where the order is unimportant (Hancock & Khoshgoftaar, 2020). In one-hot encoding, new variables are created for each unique class in the categorical variable. Those newly produced features have binary values of zeros (0) or ones (1), where zero denotes the absence of the corresponding category and one denotes its presence.

The following example demonstrates that procedure:

Table 3-4- One hot encoding

Venue	Venue_Asia	Venue_Europe	Venue_Oceania	Venue_America	Venue_Africa
Asia	1	0	0	0	0
Europe	0	1	0	0	0
Oceania	0	0	1	0	0
America	0	0	0	1	0
Africa	0	0	0	0	1
Asia	1	0	0	0	0
Oceania	0	0	1	0	0

After discarding the original "Venue" variable, the newly produced numerical variables can be used as inputs to the algorithms.

3.3.5 Data scaling

Not every variable in the dataset has the same range of values in real-world scenarios. For example, the total number of games played may be in the hundreds, and the runs scored per over could be in the range of 0-36. Standardization is one of the key strategies that fall under data scaling, which is used to address this sort of issue. After using this technique, several machine learning algorithms, like KNN, frequently exhibit good performance. The attributes are rescaled based on the following formula in the standardization to ensure that the mean is 0 and the standard deviation is 1.

$$X_{standardized} = \frac{X - mean(X)}{Standard\ deviation(X)} \quad (3.1)$$

3.3.6 Data resampling

Imbalanced datasets can cause issues such as prediction bias towards the majority class, insufficient representation of the minority class, and overfitting to the majority class. By either raising the minority class's size or decreasing the majority class's size, data resampling techniques are applied to balance unbalanced datasets.

There are some common data resampling techniques for imbalanced datasets, such as:

- Oversampling: duplicating the instances of the minority class to increase its size (e.g., Synthetic Minority Over-sampling Technique (SMOTE))
- Undersampling: removing the instances from the majority class to reduce their size (e.g., random undersampling)

Utilizing undersampling approaches can result in information loss since they remove instances from the majority class (Batuwita & Palade, 2010).

3.4 Data Visualization and Analysis

A set of plots can be drawn to identify the distribution of variables. Plotting a correlation plot is useful to determine whether multicollinearity exists in the dataset. Factor analysis can be conducted to identify the highly correlated variables group-wise. Performing Principal Component Analysis (PCA) and visualizing it in a 3D plot gives an idea of the appropriate algorithms before fitting the models (Tam, 2021).

3.4.1 Multicollinearity

Multicollinearity is defined as a significant correlation between independent variables. It is difficult to determine how each independent variable impacts the dependent variable and the importance of the independent variables when there is multicollinearity. Due to this, it can also produce inaccurate and unstable results. There are two main ways to eliminate multicollinearity: variable selection methods and modified estimator methods (Chan et al., 2022).

Correlation-association plot

A correlation matrix can be used to visually examine the correlation between all independent variables. Multicollinearity exists if there is a high degree of correlation (a correlation coefficient that is close to 1) between two or more variables. However, the correlation matrix can only be used to show how numerical variables relate to one another. Another method is required to determine the association between categorical variables. The Dython library in Python (<https://pypi.org/project/dython/>) can be used to find the correlations between numerical and categorical variables. It provides a simple and effective method for analyzing and visualizing the relationship between variables. The Dython library determines the strength of association or correlation between features in a data set by calculating:

- For continuous-continuous cases, Pearson's R
- For categorical-continuous cases, the correlation ratio
- For categorical-categorical cases, Cramer's V or Theil's U

It can be difficult to remove correlated variables or select uncorrelated variables directly by examining the correlation matrix because correlations do not necessarily imply multicollinearity (Chan et al., 2022). Also, it doesn't provide any clear details to determine which variables are the most important and which group of variables are correlated together.

Exploratory Factor Analysis

Some of the past studies have proposed variable selection methods that combine factor analysis to solve the multicollinearity problem, e.g., (Garg & Tai, 2012). Factor analysis groups highly correlated variables together into factors in order to reveal the underlying concept of a set of variables (Gie Yong & Pearce, 2013). Therefore, when reducing multicollinearity, it would be preferable to use factor analysis rather than the correlation matrix.

Using factor analysis, multicollinearity can be removed in two main ways:

1. Factor score technique: This technique creates new variables called factor scores by weighing the original variables according to their factor loadings. The factor scores then take the place of the original variables as uncorrelated independent variables. This technique successfully eliminates multicollinearity.

2. Variable reduction: Variables in the same factor are typically highly correlated. With this method, the number of variables in the study is reduced by removing some of the variables in the same factor. The feature importance plot can be utilized for that process, and only the most important variables from each factor can be retained by looking at that plot. With the use of this technique, the analysis can be made easier to understand, the interpretability can be enhanced, and multicollinearity can be reduced to a considerable extent.

In factor analysis, all the variables should be continuous or ordinal and scale. The Kaiser-Meyer-Olkin test (KMO test) and Bartlett's test are applied to check whether the dataset is appropriate for factor analysis or not. Bartlett's test examines the correlation between the observed variables. If the p-value in Bartlett's test is less than a specified significant level (e.g., less than 0.05), the correlation matrix is not an identity matrix, and factor analysis can be performed. The KMO Test evaluates if the sample size is adequate for factor analysis. Usually, if KMO is greater than 0.6, it is considered adequate (Sharma, 1995). To choose the number of factors, the Kaiser criterion and scree plot can be used, and factor rotations such as varimax can be used to improve the interpretability. The Kaiser criterion approach takes the number of eigenvalues greater than one as the number of factors.

3.4.2 Selection of suitable algorithms

Data visualization can be used to estimate suitable machine-learning algorithms for the study. A linear algorithm may be helpful when fitting models if the data have a linear, separable pattern in a binary classification problem. If the data points follow a pattern where the points are not well separated, using a nonlinear machine learning algorithm might be advantageous (Ghosh et al., 2019). A scatter plot can be used to determine whether such patterns exist. However, there are practical limitations on how many dimensions can be plotted. A scatter plot can only be used to represent two- or three-dimensional data. As a remedy, PCA can contribute to reducing the dimensions of the dataset to two or three components (Salih Hasan & Abdulazeez, 2021). Then, appropriate machine learning techniques can be proposed by examining the PCA plot.

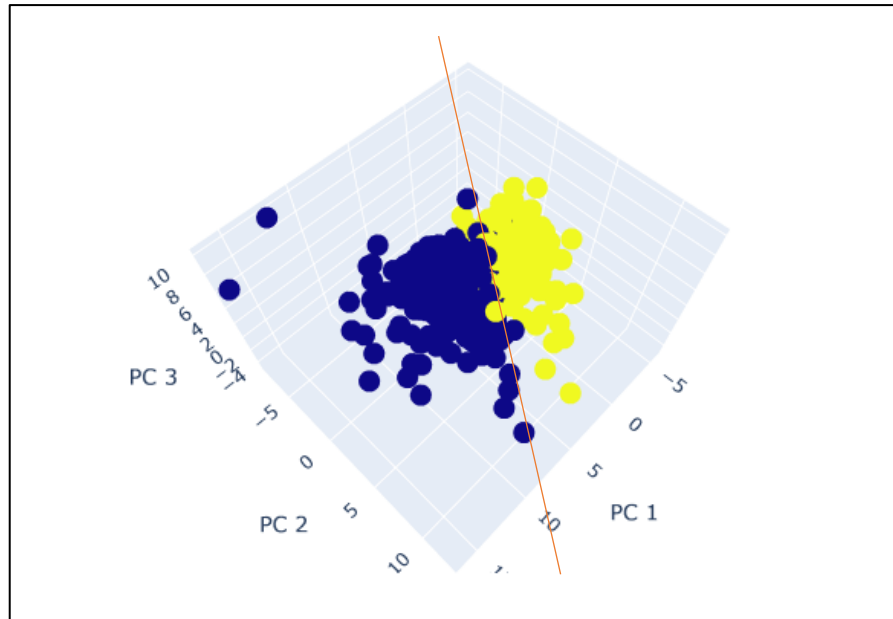


Figure 3-6- 3-D PCA plot for linearly separable data

Figure 3-6 displays that the data are separable linearly between the two classes. As in scatter plots, a linear algorithm could be used here if the data points are linearly separable. Logistic regression and Naïve Bayes are linear algorithms, while KNN, Random Forest, and XGBoost are generally non-linear algorithms. SVM can be considered both linear and non-linear algorithms. If a linear kernel is utilized in SVM, the algorithm will be linear; if not, it will be non-linear. PCA can be an excellent starting point when selecting suitable algorithms, but it cannot guarantee that the selected algorithm type will be the optimal type for the problem. Sometimes it's necessary to test several algorithms to see which one produces the best outcomes.

3.5 Model Training

3.5.1 Machine learning classification algorithms

The focus of machine learning, a subfield of artificial intelligence, is to build models by learning patterns in the data and then make predictions using those learned models. There are three main types of machine learning models:

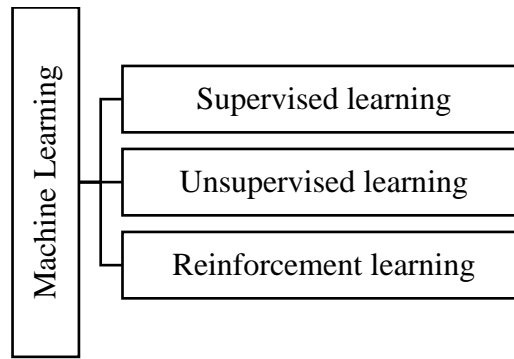


Figure 3-7- Machine learning approaches

Supervised learning involves learning the algorithm to map a given input to a desired labeled output, and it can be broadly classified into two subproblems: classification and regression, which predict categorical labels and continuous outputs, respectively.

The problem in this study is a binary classification problem. Therefore, supervised learning classification algorithms were considered.

Logistic Regression

Logistic Regression is used to represent the relationship between the predictor variables and the binary outcome, and for that, it uses a logistic function (sigmoid function), which translates the predicted values to a probability between 0 and 1. The model predicts that the instance belongs to the particular class if the estimated probability is greater than 0.5; otherwise, it predicts that it does not. As in Linear Regression, Logistic Regression models can also be regularized using L1 (i.e., lasso penalty), L2 (i.e., ridge penalty), or elastic net penalties. Regularization is a powerful technique that addresses several problems, including overfitting and multicollinearity. When a model overfits, it performs well on training data but badly on testing or unseen data. The hyperparameter "C," which is the inverse of the regularization strength in logistic regression, controls the regularization strength. The lower the value of "C," the stronger the regularization, and the higher the value of "C," the weaker the regularization (Géron, 2019, p. 149).

Support Vector Machine (SVM)

SVMs are well suited for small or medium-sized datasets rather than larger datasets. SVM classifiers, unlike logistic regression classifiers, do not provide probabilities for each class. Instead, SVMs use a margin-based approach to classify the data (Ghosh et al., 2019). Its goal is to determine the hyperplane and predict the class of new inputs by determining the side of the hyperplane. It has the capability of performing not only linear but also non-linear classification. The kernel trick comes into play when data has a non-linear relationship, and kernel functions, including the radial basis function (RBF) kernel, polynomial kernel, and sigmoid kernel, can be used to handle non-linear classification (Géron, 2019, pp. 159-163). Another important thing about SVMs is that they are very sensitive to feature scales. Therefore, SVMs perform well on scaled features.

Naïve Bayes

The algorithm's name has the prefix "Naive" since it assumes that features are independent. However, in real-life scenarios, this is frequently not the case. The name "Bayes" comes from the fact that it is a probabilistic machine-learning algorithm based on Bayes' Theorem. The probability of each class is calculated by the Naïve Bayes algorithm, and the class with the highest probability is chosen as the prediction. Naïve Bayes has different types of models such as Gaussian, Multinomial, and Bernoulli. The Gaussian Naïve Bayes algorithm assumes that variables have a normal (Gaussian) distribution, and it operates on continuous values. However, even if the data distribution marginally deviates from a normal distribution, Gaussian Naïve Bayes can produce useful results. Multinomial Naïve Bayes is used to handle count data, whereas Bernoulli Naïve Bayes is used to handle binary data.

K-Nearest Neighbors (KNN)

The KNN algorithm is a simple yet effective similarity measure-based non-parametric technique, which means it makes no assumptions about the underlying distribution of the data. It can be classified as a lazy learner because it waits to learn until a new test data point is met. A new data point is assigned to the majority class amongst its K-nearest neighbors by measuring the distance between the new data and the existing data. Different distance metrics are available to measure that distance, such as Euclidean, Manhattan, and Cosine similarity. However, there isn't a set method for figuring out the value of K, and it is advised

to choose an odd number to avoid ties (Boateng et al., 2020). The data should be scaled before using this approach, and it's vital to remember that KNN might not perform well with a large number of features. Hence, it is good to implement a dimensionality reduction technique like feature selection to select the best set of variables.

Random Forest

The fundamental principle of Random Forest is the creation of a strong model by combining several individual decision trees, which are considered weak models, using different random subsets of the data and features. Making the decision about the prediction changes based on the problem type. Random Forest can be applied to either a regression or classification problem. However, because classification is the primary goal of the study, its classification approach is primarily discussed here. The category that has the majority of votes given by decision trees is considered the final prediction of a random forest classification problem. Random forest is one of the most widely used machine learning algorithms due to its variety of advantages, such as its high accuracy, ability to handle a large number of features, and reduction of overfitting. Hyperparameter tuning and cross-validation are some of the key ways to improve the accuracy of Random Forest. This algorithm contains the functions required to easily generate feature importance. However, sometimes the Random Forest algorithm takes considerable time to execute since it considers each of the decision trees.

XGBoost

Boosting is another ensemble technique that learns from combining several weak learners. Boosting involves adding predictors to the ensemble repeatedly, with each new predictor attempting to fix the errors created by the preceding predictors until no further improvements are seen. AdaBoost (Adaptive Boosting) and Gradient Boosting are the two most popular available boosting methods (Géron, 2019, pp. 201-202). Gradient Boosting is more adaptable and powerful than AdaBoost, but if it is not correctly tuned, Gradient Boosting might overfit. While adding new models, Gradient Boosting reduces loss by employing the gradient descent algorithm. Gradient Boosting has been enhanced in XGBoost which stands for "Extreme Gradient Boosting." While XGBoost functions similarly to gradient boosting, it differs from classic gradient boosting in a number of significant ways. XGBoost offers the following key features that make it more efficient:

- Overfitting is avoided via XGBoost's capability to halt tree growth when the loss function falls below a predetermined threshold.
- Decision tree construction may be parallelized via XGBoost across many cores and even several machines.
- It has built-in regularization, which helps avoid overfitting.
- It has the ability to handle missing values without any pre-processing steps.
- It offers measures of feature importance, which are useful for feature selection.

The quick execution and high performance of this algorithm are the two main reasons for its popularity.

Voting Classifier

In most cases, the voting classifier can improve accuracy by aggregating the predictions of individual classifiers such as logistic regression, Naïve Bayes, Random Forest, and others. The voting classifier comes in two types: hard voting and soft voting. In hard voting, predictors give predictions based on the majority vote. For example, if four classifiers predicted their outputs as 1, 2, 1, and 1, most of the classifiers would predict the output as 1. According to that result, the voting classifier with hard voting selects 1 as the final prediction. On the other hand, soft voting considers class probabilities. If you have 0.55, 0.45, 0.61, and 0.43 for class 1 probabilities and 0.32, 0.39, 0.22, and 0.25 for class 2 probabilities, then the average probability for class 1 is 0.51 and the average probability for class 2 is 0.295. Based on that result, the voting classifier with soft voting selects class 1 as the final prediction since it has the highest average value. In general, soft voting achieves better results than hard voting because of the consideration of weights (Géron, 2019 p. 194).

3.5.2 Feature Importance

Feature importance plots visually represent the importance of each feature on the target variable in the classification prediction problem. Using this measurement, the most important features of a particular problem can be identified. There are many ways to calculate the features' importance. The models that are based on trees, like Random Forest and XGBoost, calculate feature importance measurement by considering the number of times a feature is used to split the data. The feature is deemed more important if there are more splits. In linear models like logistic regression, the magnitude of the feature

coefficients can be used to measure how important a feature is. A larger magnitude indicates a substantial influence on the target variable. This technique may also be used to build the SVM feature importance plot. However, this is only true for SVM when a linear kernel is employed. This approach is not applicable to SVMs with non-linear kernels.

3.5.3 Hyperparameter tuning

Hyperparameters are settings made prior to the model's training. Tuning those parameters to obtain the best results is one of the important steps in the process of training a machine learning model. Grid search is one of the ways to perform hyperparameter tuning. In a grid search, a collection of hyperparameters and their potential values are defined, and the algorithm selects the best set of possible values for each hyperparameter after considering all possible combinations (Yang & Shami, 2020).

3.5.4 K-fold cross-validation

The data set is randomly divided into K numbers of folds, often referred to as subsets, in K-fold cross-validation. Usually, ten or five is chosen as the K value, which results in 10-fold cross-validation or 5-fold cross-validation, respectively. After deciding the number of folds, the K-1 number of folds is taken as the training set, and the other fold is used as the validation set. This process repeats K times, taking only one fold as the validation set at each iteration. The final result is obtained by averaging the outcomes of each step once all iterations have been completed. These folds are mostly the same size but can vary slightly. In cases like that, the weighted average is preferable to the average. K-fold cross-validation helps in both model hyperparameter optimization and overcoming the overfitting issue (Charilaou & Battat, 2022).

3.6 Model evaluation

3.6.1 Classifier performance

For the given classifier and test set, there are four possible outcomes for a binary classification problem. Those four outcomes can be represented by a two-by-two confusion matrix.

		Predicted Class	
		Negative	Positive
Actual Class	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

Figure 3-8- The confusion matrix

- True Negatives (TN) represent the number of instances that are correctly classified as negative.
- False Negatives (FN) represent the number of instances that are incorrectly classified as negative.
- False Positives (FP) represent the number of instances that are incorrectly classified as positive.
- True Positives (TP) represent the number of instances that are correctly classified as positive.

To compare the classification models, a way to measure the classification performance is needed. Different kinds of scores are available for this purpose, and those scores are based on the confusion matrix and Receiver Operating Characteristics (ROC) curve.

Scores based on Confusion Matrix	Score based on ROC curve
<ul style="list-style-type: none"> • Accuracy • Recall • Precision • F1 score 	<ul style="list-style-type: none"> • The area under the ROC curve (ROC AUC)

Figure 3-9- Classification metrics

Accuracy

The proportion of correct predictions made by a model relative to the total number of predictions is called accuracy. When the classes are balanced, which means there are nearly equal numbers of positive and negative cases in the dataset, accuracy can be a useful metric.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

(3.2)

Precision

The precision, or positive prediction value, calculates the percentage of correctly predicted positive outcomes. It is an important metric when the cost of false positive predictions is high, but it should not be used alone and should be considered alongside other metrics.

$$Precision = \frac{TP}{TP + FP}$$

(3.3)

Recall

Recall, also referred to as sensitivity, true positive rate, or hit rate, quantifies the percentage of positive events that the model successfully and accurately recognizes. In situations where the cost of false negatives is high, recall is especially helpful. Recall should be used in combination with other measures, just like precision.

$$Recall = \frac{TP}{TP + FN}$$

(3.4)

F1 score

The F1 score, also known as the harmonic mean of precision and recall, is a helpful classification metric that offers a balance between precision and recall. It can provide a more complete view of a model's performance compared to precision or recall alone.

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

(3.5)

ROC-AUC

The relationship between the true positive rate (recall) and the false positive rate (1 - specificity) of a classifier at different threshold values is shown graphically by the ROC curve. The performance of a classifier is numerically quantified using the AUC (i.e., area under the ROC curve).

The ROC-AUC and the F1-score are both useful when there is a class imbalance, but the performance metric "Accuracy" can lead to misleading situations (Ul Mustafa et al., 2017). Additionally, as was already mentioned, relying just on recall and precision is not good. A model with high precision but low recall, or a model with low precision but high recall, can have low overall performance. To solve that problem also, the F1 score can be considered.

3.6.2 Partial Dependence Plot (PDP)

The marginal impact of one or two variables on the outcomes predicted by the machine learning model can be displayed by presenting partial dependence plots (PDPs/PD plots) (Friedman, 2001). Usually, it illustrates how a change in the values of a specific predictor variable affects the target variable when other predictor variables are held constant. Partial dependency plots can be considered one of the better techniques to get some insights from complicated models, and these plots can also be used to check whether the model follows patterns that are actually there.

Chapter 4

Datasets

The datasets were created by considering 519 games played between the top 9 teams (as of October 14, 2022) from the 1st of January 2011 to the 14th of October 2022. The study only considered games with a clear outcome (won or lost); tied games, matches with no results, and abandoned matches were excluded. Initially, the plan was to take the top ten teams, but after looking at how many matches each side had played, it was found that Afghanistan had only played 29 games, which is a very low count compared to the other teams in the top 10. Therefore, only the top nine teams—India, England, West Indies, Pakistan, Australia, Sri Lanka, South Africa, New Zealand, and Bangladesh—were taken into consideration for the study. Additionally, data from the ICC T20 World Cup 2022, which took place in Australia from the 16th of October to the 13th of November, as well as pre-match betting odds for those matches, were collected with the aim of evaluating the effectiveness of the proposed methodology.

Table 4-1- Number of matches played



POS	TEAM
1	India
2	England
3	Pakistan
4	South Africa
5	New Zealand
6	Australia
7	West Indies
8	Sri Lanka
9	Bangladesh
10	Afghanistan

Figure 4-1-Men's T20I team rankings as of Oct 14, 2022

Team	Number of matches
India	139
Pakistan	131
West Indies	127
England	123
Australia	122
Sri Lanka	120
South Africa	111
New Zealand	108
Bangladesh	86
Afghanistan	29

Two data sources—historical data and Twitter data—were considered when meeting the objectives. Three main datasets were built. The first one contained historical data related to past matches. The second one was built using the features extracted from the collected tweets. The last one is a combination of both the first two datasets. From here onwards, these three datasets are respectively known as the "Historical," "Twitter," and "Historical+Twitter" datasets. In addition to that, the odds of bookmakers were also collected for T20 World Cup matches in 2022, as mentioned.

The features were chosen to include in the datasets based on earlier research studies, and this chapter discusses the steps used to generate these features.

4.1 Historical Dataset

4.1.1 ESPNCricinfo's Statsguru

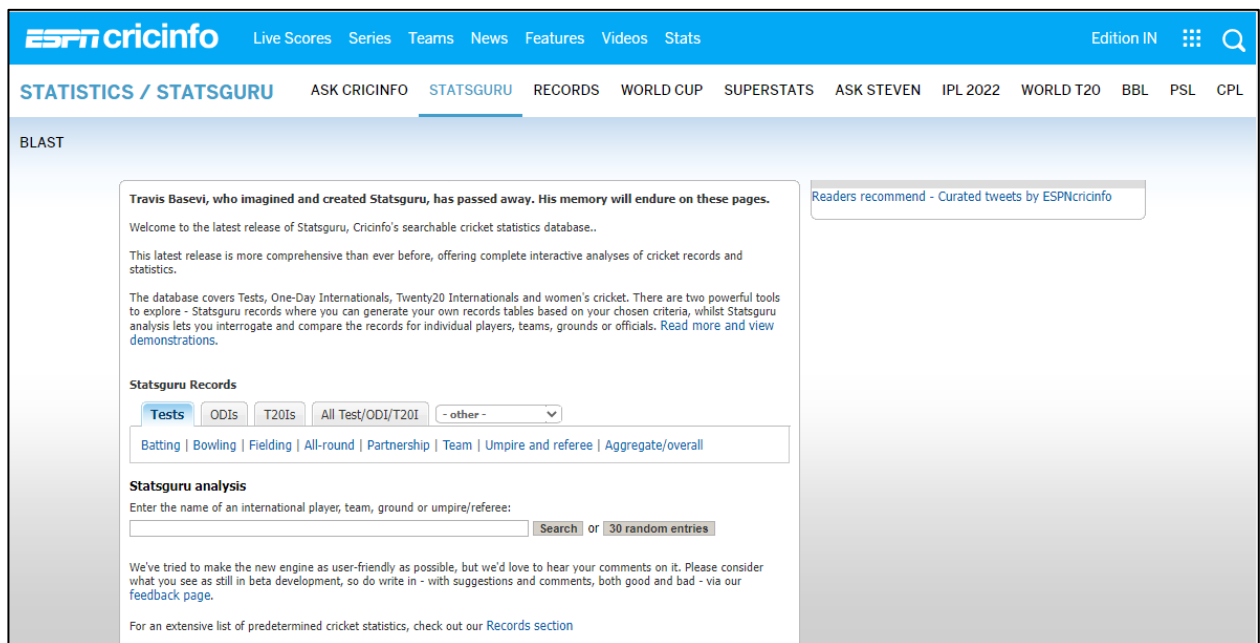


Figure 4-2- Cricinfo's Statsguru

The historical match data required for this research were obtained from Statsguru (<https://stats.espncricinfo.com/ci/engine/stats/index.html>), a cricket statistics database accessible through ESPNCricinfo. This database includes data on player and team accomplishments, records, and rankings for all three international formats (i.e., One-Day Internationals, Twenty20 Internationals, and Tests), women's cricket, and youth cricket.

Statsguru allows anyone to create unique record tables based on their own criteria. Additionally, it has the ability to analyze and compare the stats for specific players, teams, venues, or even umpires.

Before the game, each team's batting, bowling, and team records were taken into consideration in order to generate the historical dataset for this study. For that, three separate queries had to be submitted before each match. Since Statsguru presents the findings in tabular form, it is necessary to use a web scraping approach to retrieve the relevant data from these output tables. The data on this website are well organized, so only the removal of unnecessary columns is required.

4.1.2 Scraping historical data

The data needed for this research were simply gathered using the Pandas package because Statsguru provides data in tabular format. The web page containing the desired study results was primarily made up of six <table> tags. A list of data frames was generated after the URL of the result page was passed to the read_html() function. The data needed to develop the historical dataset was contained in the third <table> tag. Therefore, the necessary data were gathered by extracting the third item (the second index) from the list.

Tests ODI **T20Is** All Test/ODI/T20I - other -

Batting | Bowling | Fielding | All-round | Partnership | Team | Umpire and referee | Aggregate/overall

View overall figures [change view]
Primary team South Africa ☒
Opposition team Afghanistan ☒ or Australia ☒ or Bangladesh ☒ or England ☒ or India ☒ or New Zealand ☒ or Pakistan ☒ or South Africa ☒ or Sri Lanka ☒ or West Indies ☒
Start of match date less than or equal to 8 Jan 2011 ☒
Match result won match ☒ or lost match ☒
Grouped by team ☒
Ordered by runs scored (descending)

Page 1 of 1 Showing 1 - 1 of 1 First Previous Next Last Return to query menu Cleared query menu

Team	Players	Span	Mat	Inns	NO	Runs	HS	Ave	BF	SR	100	50	0	4s	6s
South Africa	47	2005-2010	34	266	62	4541	94	22.25	3835	118.40	0	20	24	385	157

Page 1 of 1 Showing 1 - 1 of 1 First Previous Next Last Go to page Go

Statsguru includes the following current or recent Twenty20 Internationals:
 India v New Zealand at Ahmedabad, 3rd T20I, Feb 1, 2023 [T20I # 1992]
 India v New Zealand at Lucknow, 2nd T20I, Jan 29, 2023 [T20I # 1991]
 India v New Zealand at Ranchi, 1st T20I, Jan 27, 2023 [T20I # 1990]

Feedback Print

Figure 4-3- An output results page of Statsguru

```

▶ <form action style="margin:0px; padding:0px;">...</form>
▶ <div class="guruNav" style="padding-top:0px;">...</div>
▶ <table class="engineTable" style="margin:0px;">...</table>
▶ <table class="engineTable" style="margin-bottom:5px;">...</table>
▼ <table class="engineTable">
  <caption>Overall figures</caption>
  <thead>
    ▶ <tr class="headlinks">...</tr>
  </thead>
  <tbody>
    ▼ <tr class="data1">
      ▶ <td class="left" nowrap="nowrap">...</td>
      <td>47</td>
      <td class="left" nowrap="nowrap">2005-2010</td>
      <td>34</td>
      <td>266</td>
      <td>62</td>
      ▶ <td>...</td>
      <td class="padAst">94</td>
      <td>22.25</td>
      <td>3835</td>
      <td>118.40</td>
      <td>0</td>
      <td>20</td>
      <td>24</td>
      <td>385</td>
      <td>157</td>
      ▶ <td class="padDD">...</td>
    </tr>
  </tbody>
</table>

```

Figure 4-4- HTML code of a Statsguru results page

4.1.3 Formulating the initial historical dataset

The above two snapshots demonstrate how to obtain Team South Africa's previous batting statistics for their game against India on January 9, 2011. The South African team's bowling and team records were obtained using the same method. The same procedure was applied to the Indian team as well. Likewise, batting, bowling, and team records for the previous x matches played by each team against the top ten teams were taken into consideration prior to any match.

Each instance of the historical dataset represented a match, and the instance was mainly split into three parts: Team1 features, Team2 features, and the target variable. In addition, the dataset also contained the venue variable. This study was to predict the outcome of a game as a win for Team1 or a win for Team2. Most studies in this field consider the home team or the away team as either Team1 or Team2. But this study was for international

matches and contained data not only for the bilateral series. Asia Cup matches, World Cup matches, and other tri-series were also considered. Therefore, matches were away matches for most of the teams in those cases. Therefore, in this study, another way was taken into consideration. Here, Team 1 is the team that has the highest win/loss ratio for past matches of the two teams in each match. Team 2 is the team that has the lowest win-loss ratio. This win/loss value was directly extracted from the Statsguru database, and it can be defined as follows:

$$\text{Win/Loss ratio} = \frac{\text{Number of won games}}{\text{Number of lost games}} \quad (4.1)$$

After removing unnecessary variables, the dataset for the 519 matches with the following variables was saved in local files for later use.

Table 4-2- ESPNcricinfo extracted features

Variable	Description
Date	Date of the match played
Venue	Match played ground
T1_Mat	Total number of matches played by Team1
T1_BallsBatted	Total number of balls faced by Team1
T1_OversBowled	Total number of overs bowled by Team1
T1_RunsScored	Total runs scored by Team1
T1_4s	Total number of 4s earned by Team1
T1_6s	Total number of 6s earned by Team1
T1_RunsConceded	Total runs conceded by Team1
T1_WktsTaken	Total number of wickets taken by Team1
T1_W/L	Win/Loss ratio of Team1
T1_WktsLost	Total number of wickets lost by Team1
T2_Mat	Total number of matches played by Team2
T2_BallsBatted	Total number of balls faced by Team2
T2_OversBowled	Total number of overs bowled by Team2
T2_RunsScored	Total runs scored by Team2

T2_4s	Total number of 4s earned by Team2
T2_6s	Total number of 6s earned by Team2
T2_RunsConceded	Total runs conceded by Team2
T2_WktsTaken	Total number of wickets taken by Team2
T2_W/L	Win/Loss ratio of the Team2
T2_WktsLost	Total number of wickets lost by Team2
Winner	Winner of the match (1 indicates Team1, 0 indicates Team2)

4.1.4 Formulating the final historical dataset

The following new features were created with the help of the variables in the above-mentioned dataset:

Venue

As the venue, Cricinfo Statsguru provides the names of the grounds for each match. In order to make that variable more useful and approachable, these ground names were categorized by continent: Asia, Africa, Europe, America, and Oceania.

Average Runs Scored Per Over

$$Tj_AvgRunsScored = \frac{\sum_{i=1}^n Runs\ scored_i^j}{\sum_{i=1}^n Number\ of\ overs\ batted_i^j} \quad (4.2)$$

The "Tj_AvgRunsScored" variable indicates the average runs scored per over by Team j. This variable was created by dividing the sum of the total runs scored by Team j in the last n matches by the total number of overs batted by Team j in the previous n matches. However, the number of overs batted was not directly available in the Cricinfo database. Therefore, that variable was calculated using the number of balls faced variable, as shown in Appendix Figure B-1.

Average Runs Conceded Per Over

$$Tj_AvgRunsConceded = \frac{\sum_{i=1}^n Runs\ conceded_i^j}{\sum_{i=1}^n Number\ of\ overs\ bowled_i^j} \quad (4.3)$$

This variable denotes the average runs conceded per over by Team j. By dividing the total runs conceded by Team j in the last n matches by the total overs bowled by Team j in the previous n matches, this variable was created.

Average Boundaries Earned Per Over

$$Tj_AvgBound = \frac{\sum_{i=1}^n (4s\ earned + 6s\ earned)_i^j}{\sum_{i=1}^n Number\ of\ overs\ batted_i^j} \quad (4.4)$$

The average number of boundaries earned per over by Team j is denoted by this variable. This variable was calculated by dividing the total number of boundaries (the sum of 4s and 6s) earned by Team j in the previous n matches by the total number of overs batted by that team in the previous n matches.

Average Wickets Taken Per Over

$$Tj_AvgWktsTaken = \frac{\sum_{i=1}^n Number\ of\ wickets\ taken_i^j}{\sum_{i=1}^n Number\ of\ overs\ bowled_i^j} \quad (4.5)$$

The variable “Tj_AvgWktsTaken” was created to denote the average number of wickets taken per over by Team j. This variable was derived by dividing the total number of wickets that Team j has taken in the previous n matches by the total number of overs that Team j has bowled in those same matches.

Average Wickets Lost Per Over

$$Tj_AvgWktsLost = \frac{\sum_{i=1}^n \text{Number of wickets lost}_i^j}{\sum_{i=1}^n \text{Number of overs batted}_i^j} \quad (4.6)$$

This variable represents Team j's average number of wickets lost per over. This variable was created by dividing Team j's total number of wickets lost in the last n matches by the total number of overs batted by that team in the last n matches.

Team j here stands in for either Team 1 or Team 2. In addition to these variables, some variables from the initial dataset, which were discussed in the previous subsection, were added to the final dataset, and the features of the final dataset were as follows:

Table 4-3- Finalized historical features

Variable	Description
Venue	Match played continent
T1_Mat	Total number of matches played by Team1
T1_AvgRunsScored	Average runs scored per over by Team1
T1_AvgBound	Average number of boundaries earned per over by Team1
T1_AvgRunsConceded	Average runs conceded per over by Team1
T1_AvgWktsTaken	Average number of wickets taken per over by Team1
T1_W/L	Win/Loss ratio of Team1
T1_AvgWktsLost	Average number of wickets lost per over by Team1
T2_Mat	Total number of matches played by Team2
T2_AvgRunsScored	Average runs scored per over by Team2
T2_AvgBound	Average number of boundaries earned per over by Team2
T2_AvgRunsConceded	Average runs conceded per over by Team2

T2_AvgWktsTaken	Average number of wickets taken per over by Team2
T2_W/L	Win/Loss ratio of Team2
T2_AvgWktsLost	Average number of wickets lost per over by Team2
Winner	Winner of the match (1 indicates Team1, 0 indicates Team2)

The final "Historical" dataset contained 519 observations and 16 variables.

4.2 Twitter dataset

This section describes the steps that were followed during the creation of the “Twitter” dataset. There were two primary phases in the process of creating this dataset. Tweets were scraped in the first phase, and then features were extracted from those scraped tweets in the second phase.

4.2.1 Scraping tweets

The Snsrape library was used to scrape the tweets, and the employed search query for that had four main inputs:

1. Hashtags related to a particular match
2. User handles related to a particular match
3. Language of tweets
4. Dates and time range

Search query

To fill the first two inputs, a list of hashtags and user handles associated with each team, depending on familiarity, was manually created and verified using search queries on Twitter. These user handles were extracted from each team's official Twitter accounts.

Table 4-4 Hashtags and handles list

Team	Hashtags	Handles
India	#TeamIndia #MenInBlue #BCCI #IndianCricket #IndianCricketTeam #bharatarmy	@BCCI
England	#ECB #englandcricket #cricketengland #englandteam #EnglandCricketTeam	@englandcricket
Pakistan	#BackTheBoysInGreen #TheGreenArmy #packistancricket #packistancricketteam #TheRealPCB #PakCricket #cricketpakistan #PCB	@TheRealPCB
South Africa	#ProteaFire #Proteas #PureProtea #southafricacricket #CricketSouthAfrica #rsacricket #sacricket #SouthAfricanCricket #CSA	@ProteasMenCSA
New Zealand	#BACKTHEBLACKCAPS #CricketNation #blackcaps #NZC #newzealandcricket #nzcricket	@BLACKCAPS
Australia	#australiancricket #CricketAustralia #australiacricket #CricketAus	@CricketAus
West Indies	#MenInMaroon #WiAllin #WestIndies #Windies #windiescricket #westindiescricket	@windiescricket
Sri Lanka	#OneTeamOneNation #RoaringForGlory #ApeKollo #TeamSriLanka #SriLankanCricketTeam #SriLankanTeam #SrilankaCricket #GemmakThamai #OfficialSLC	@OfficialSLC
Bangladesh	#RiseOfTheTigers #BCB #BCBTigers #BangladeshCricket #bangladeshcricketteam #bdcricet #bdtigers #bdcricet_team	@BCBTigers

The following types of hashtags were also taken into account when scraping the tweets to expand the dataset:

#Team1vTeam2 #Team1vsTeam2 #Team2vTeam1 #Team2vsTeam1

Tweets with these types of hashtags were categorized according to the first team mentioned.

Table 4-5- Hashtags that mentioned both teams

Hashtag	Team
#Team1vTeam2 #Team1vsTeam2	Team1
#Team2vTeam1 #Team2vsTeam1	Team2

Here, Team1 and Team2 represent the shortened names of each team.

Table 4-6- Shortened names of each team

Team	Shortened name/s
India	IND
Pakistan	PAK
West Indies	WI
England	ENG
Australia	AUS
Sri Lanka	SL
South Africa	SA, RSA
New Zealand	NZ
Bangladesh	BAN

When considering the third input of the search query, English was taken as the preferred language for this study due to the limitations of other languages.

For this study, tweets that were posted between 24 hours and an hour prior to the game were considered. The fourth input to the query was added to include that specific time range. The starting time of each previous match was manually extracted using the archived schedule reports provided by Cricbuzz (<https://www.cricbuzz.com/cricket-scorecard-archives>), and the relevant timestamps utilizing that time were noted down. The times that were retrieved from cricbuzz.com were in GMT format. For the search query, the time needs to be entered in the Epoch time format. By using the online Epoch & Unix

Timestamps Conversion Tool, Epoch Converter (<https://www.epochconverter.com/>), GMT timestamps were changed into Epoch timestamps.

Tweets cleaning

After creating a proper search query with those four inputs, tweets were scraped, those tweets were converted to lowercase, and duplicates were removed. There is no benefit to this kind of task from tweets that raise a question. With the use of a regular expression pattern, a function was defined to recognize those kinds of tweets as follows:

```
def has_a_question(tweet):
    # a regular expression pattern to identify question mark
    pattern1 = r'\?'
    # a regular expression pattern to identify any question word
    pattern2 = r"\b(who|what|when|where|why|how)\b"
    # Combine the two patterns using the OR (|) operator
    pattern = r'({}|{})'.format(pattern1, pattern2)
    # search function to find tweets with question
    q_tweet = re.search(pattern, tweet, re.IGNORECASE)
    # Return True if the tweet has a question
    return q_tweet is not None
```

Figure 4-5- The code snippet to identify questions

The | operator, which acts as an "OR" operator in regular expressions, can be used to combine multiple patterns, and it is allowed to match either one of the patterns or both of them. Here, the | operator was used to combine pattern 1 and pattern 2. While questions were often presented with a question mark, this wasn't the case for the majority of tweets. For that reason, to identify the tweets that contained questions, a second pattern was defined using question terms like "why," "who," and "what." After that, by applying this function, tweets that raised a question were removed from the initial dataset.

There were still some unrelated tweets even after performing numerous filtering processes. For example, women's cricket matches or under-19 cricket matches were played between the same two nations on the same day in parallel to the men's T20I matches. Using the Excel filter tool, tweets about those matches were removed with the help of specific keywords like "U-19" and "women."

The dataset analysis also revealed that the majority of tweets contained several hashtags. Studies from the past have stated that Twitter may have wanted to emphasize the first hashtag more, as described in Chapter 2.

Therefore, tweets with multiple hashtags and handles were classified using the function below according to the first hashtag or first handle:

```
def select_tweet_with_item(tweets, list1):
    selected_tweets = []
    for tweet in tweets:
        for item in list1:
            if item in tweet:
                # Use a regular expression to find all hashtags & handles mentioned in the tweet
                hashtags = re.findall(r"#\w+", tweet)
                handles = re.findall(r"@w+", tweet)
                # Check if the specified hashtag or handle appears before any other hashtags or handles in the tweet
                if (item in hashtags and hashtags.index(item) == 0) or (item in handles and handles.index(item) == 0):
                    selected_tweets.append(tweet)
                    break
    return selected_tweets
```

Figure 4-6 -The code snippet to classify tweets with multiple hashtags

A tweet was considered a Team1 tweet if the first hashtag in the tweet was related to Team1. Otherwise, if the first hashtag in a tweet is related to Team2, that tweet was classified as a Team2 tweet. If the tweet has a user handle in its first index, the same procedure was followed. These tweets were saved locally as CSV files for each team for each match.

4.2.2 Extracting basic features from tweets

Then, using these scraped tweets, the following important information was extracted as the second phase of generating the “Twitter” dataset:

- Total number of tweets of a team for a match
- Count of positive tweets of a team for a match
- Count of tweets with the pattern “Team j win” or “Team j will win” of a team for a match

Here, Team j represents the team names or the shortened names of each team. By simply creating a list of possible patterns for each team, the possibility of having either one of the above-mentioned patterns was checked

Running each file to obtain values for these variables was a difficult task. Therefore, using Streamlit, a web application was developed to automate that process. This web application's primary goal was to determine a tweet's sentiment. However, this application was expanded to obtain the values for each basic variable for each team for a specific match.


Cric-Tweets

Analyze Tweet

Tweet here:

Analyze CSV

Upload file:

 Drag and drop file here
Limit 200MB per file

Browse files

Figure 4-7- The web application

This application has two sections. Using the first section, you can find the sentiment of a single tweet.

Analyze Tweet

Tweet here:

Come on boyssss You can do it!

Label: Positive

Figure 4-8- Application section to identify the sentiment of a tweet

Using the second section, you can find the sentiment of a set of tweets saved in CSV format and also download the file with sentiment results for each tweet. In addition, the values of the basic variables are available in this section.

Drag and drop file here

Limit 200MB per file

Browse files

SA-2011-01-09.csv 1.8KB

×

	Sentiment
0 e. see you game time #savsind . will let you know about #iplauction till then. enjoy!	1
1 r your contribution to #sacriccket & contributing to uniting #sa through sport. we salute you!	1
2 i 2day playin his last game as #proteas w@ a legend	1
3 ting in de morning effort! church was gud. looking forward to the #proteas kicking some ass agai	1
4 n durban and i'm feeling much better... come on #proteas	1

Download

Winner prediction count: 0

Number of Positive Tweets: 12

Total Number of Tweets: 12

Figure 4-9- Application section to obtain sentiment and values for variables

Sentiment value 1 indicates a positive tweet, and 0 indicates a negative tweet.

These values for each variable were saved in a CSV file. Like in the "Historical" dataset, in this dataset also, a record was mainly split into three parts: Team1 features, Team2 features, and the target variable. The target variable was Winner, which could be either Team1 or Team2. Team 1 and Team 2 were also defined as the same as in the "Historical" dataset.

4.2.3 Formulating the “Twitter” dataset

The following features were calculated for the final "Twitter" dataset using the aforementioned variables: These characteristics were also based on earlier studies, and they used Twitter volume, fans' sentiment scores, and average fans' predicted scores as variables, as was described in Chapter 2. In this study, the first two features—Twitter volume and fans' sentiment score—were taken into account. Past studies calculated the fans' predicted

scores by considering the tweets containing predicted scores. However, after going through the collected tweets for this study, it was noticed that only a very small number of tweets predicted each team's final score. So, in this study, tweets with written predictions were considered instead of tweets with predicted scores. Here, for each variable, the "Tj" denotes the team, Team 1 or Team 2, and the "i" denotes the match.

Twitter Volume

$$Tj_TwitterVol^i = \frac{\text{Count of tweets}_j^i}{\text{Total number of tweets}^i} \quad (4.7)$$

The variable $Tj_TwitterVol^i$ indicates the pre-match tweet volume of Team j for i^{th} match. This variable was calculated by dividing the count of tweets of Team j for i^{th} match by the total number of tweets posted for that particular match.

Fans' sentiment score

$$Tj_FansSent^i = \frac{\text{Count of positive tweets}_j^i}{\text{Total number of tweets}^i} \quad (4.8)$$

This variable indicates the sentiment score of Team j for i^{th} match, and this variable was created by dividing the count of positive tweets of Team j for i^{th} match by the total number of tweets posted for the i^{th} match.

Fans' prediction score

$$Tj_FansPred^i = \frac{\text{Count of tweets with selected patterns}_j^i}{\text{Total number of tweets with selected patterns}^i} \quad (4.9)$$

The above variable was calculated by dividing the count of tweets with the selected patterns of Team j for i^{th} match by the total number of tweets with those selected patterns for the match.

The final Twitter dataset included the following features:

Table 4-7- Finalized Twitter features

Variable	Description
T1_TwitterVol	Volume of pre-match tweets for Team 1
T1_FansSent	Aggregated fans' sentiment for Team 1
T1_FansPred	Average mentions of Team 1 as the winner
T2_TwitterVol	Volume of pre-match tweets for Team 2
T2_FansSent	Aggregated fans' sentiment for Team 2
T2_FansPred	Average mentions of Team 2 as the winner
Winner	Winner of the match (1 indicates Team1, 0 indicates Team2)

This final "Twitter" dataset contained 519 observations and 7 variables.

4.3 Historical+Twitter dataset

The first two datasets, "Historical" and "Twitter," were merged to create the "Historical+Twitter" dataset. The variables in this dataset are therefore a combination of variables from the previous two datasets. Therefore, the "Historical+Twitter" dataset contained 519 observations and 22 variables.

Snapshots of all three datasets are given in Appendix A.

4.4 Betting odds

On a daily basis, one hour before the start of each T20 World Cup 2022 match, the pre-match betting odds were manually gathered. Just 14 of the matches contested amongst the top 9 teams had a clear winner, while the majority of the games in this tournament were called off due to the bad weather. These betting odds were obtained from [oddsportal.com](https://www.oddsportal.com/) (<https://www.oddsportal.com/>), a website that provides average or highest decimal betting odds from a variety of bookmakers. For this study, average betting odds were taken into account. The following snapshot provides further details about this website.

[Home](#) » [Cricket](#) » [World](#) » T20 World Cup Odds

My Leagues (1)

T20 World Cup (1)

→ Manage My Leagues

Search

team / player

SPORTS

SOCCER

BASKETBALL

BASEBALL

HOCKEY

TENNIS

T20 World Cup Betting Odds

NEXT MATCHES

RESULTS

STANDINGS

Cricket » World » T20 World Cup

Today, 13 Nov - Play Offs

08:00 Pakistan - England

1

2

B's

2.33

1.60

15

→ Remove "T20 World Cup" from My Leagues

T20 World Cup page help: Odds Portal lists all upcoming T20 World Cup cricket matches played in World. "B's" column indicates number of bookmakers offering T20 World Cup betting odds on a specific cricket match. Columns 1, X and 2 serve for average/biggest T20 World Cup betting odds offered on home team to win, draw and away team to win the T20 World Cup match. The top line of upcoming matches table (Cricket - World - T20 World Cup) lets you click-through to higher categories of Odds Portal betting odds comparison service.

My Coupon

No bets selected yet. To add a bet click the odds while browsing through OddsPortal!

→ Show my saved coupons (0)

World (Cricket)

T20 World Cup (1)

One Day International (1)

Twenty20 International (1)

Twenty20 International Women (1)

→ World

Betting Tools

Dropping Odds

Blocked Odds

Figure 4-10- oddsportal.com

Chapter 5

Data Analysis

Important results from the study's analysis of the data are presented in this chapter, and these significant findings were mainly used to carry out research successfully. The data analysis for this study was done on the training set. Mainly the results of the correlation-association plots, exploratory factor analysis, and PCA plots, which were carried out to identify highly correlated variables and suitable machine learning algorithms, are shown in this chapter as the important results of the analysis. To perform data analysis, the following Python libraries and modules were used:

- sklearn
- dython
- plotly
- matplotlib
- seaborn
- factor_analyzer

Distribution of the target variable

The target variable of all three main datasets, "Historical", "Twitter", and "Historical+Twitter", is the "Winner," either Team 1 or Team 2. As mentioned in the previous chapter, Team 1 has the highest win/loss ratio of the two teams. Team 2 is the team that has the lowest win/loss ratio between the two teams. A bar plot can be used to show the target variable's distribution.

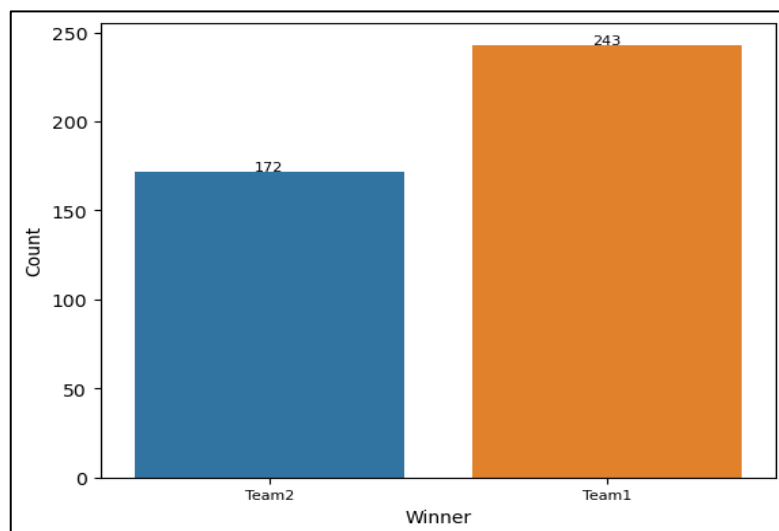


Figure 5-1- Bar plot for the target variable's distribution

Figure 5-1 shows that the target variable is not well balanced. Therefore, this plot suggests that when building models, it is important to be aware of the issues generated by unbalanced datasets and apply appropriate techniques to prevent them, as discussed in Chapter 3.

5.1 Data Analysis – Historical dataset

A correlation-association plot provided by the Dython library, which considers both numerical and categorical data, can be used to get a primary idea about multicollinearity. Following is the plot for independent variables in the Historical dataset.

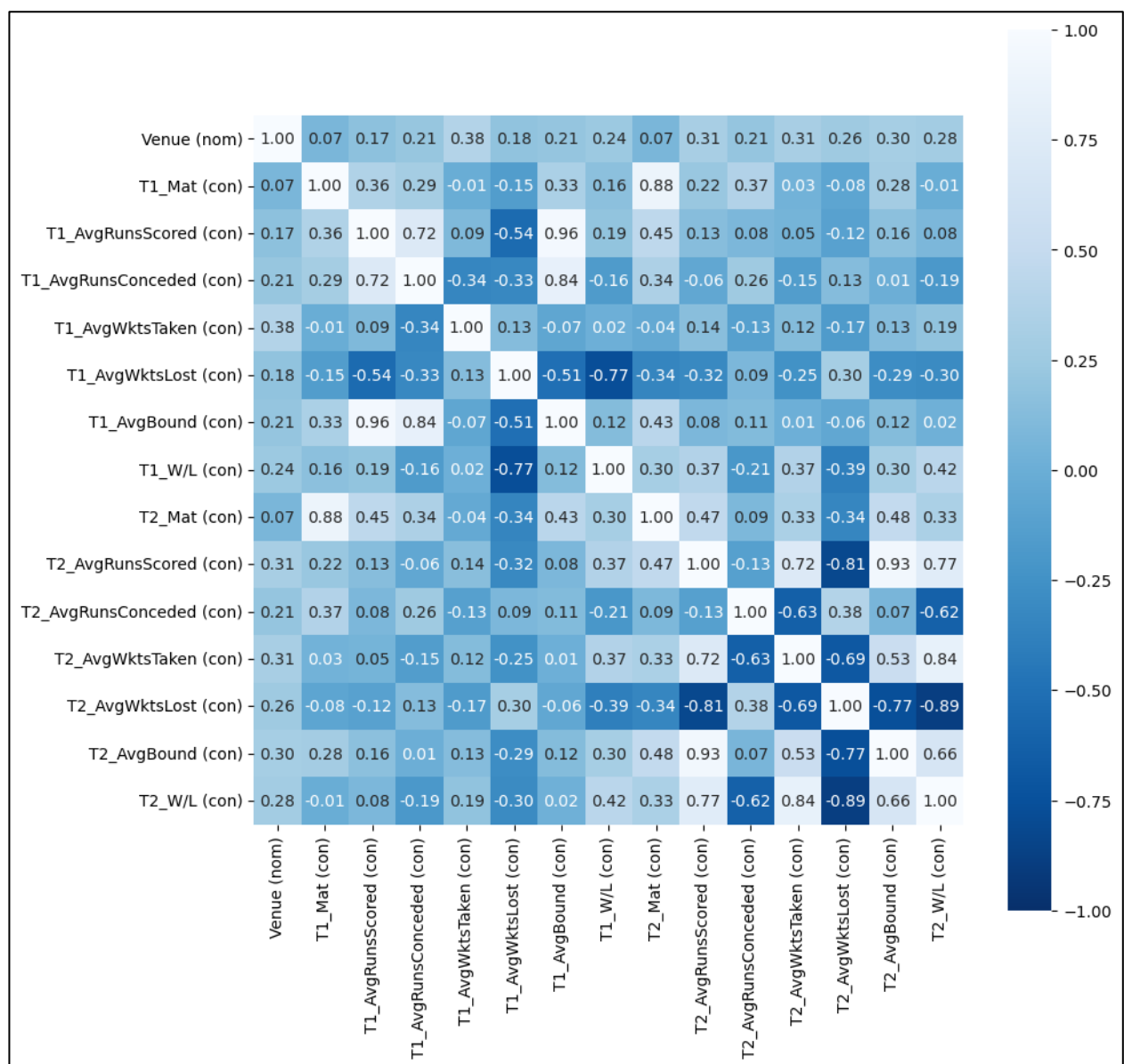


Figure 5-2 -Correlation-association plot for the Historical dataset

The plot shows that there is a heavy correlation between many variables. For example, there are strong positive correlations (0.96 & 0.93) between the average number of boundaries scored per over and the average runs scored per over by each team. There is a strong negative correlation (-0.89) between the average wickets lost by Team 2 (T2_AvgWktsLost) and the win/loss ratio of Team 2 (T2_W/L). These two are only a few of the heavy correlations that exist in the dataset. From these findings, it can be concluded that multicollinearity exists in this dataset.

Next, the results of the factor analysis are discussed. The factor analysis was carried out to identify the highly correlated variables group-wise and to provide a solution for the multicollinearity. Both Bartlett's test and the KMO test showed that there is no problem conducting a factor analysis for this dataset. Bartlett's test p-value was 0, which means the test is statistically significant and indicates that the observed correlation matrix is not an identity matrix. The overall KMO for the data was 0.68, which is greater than 0.6. That indicates that the planned factor analysis can proceed without any issues. Then, a scree plot was drawn to choose the number of factors.

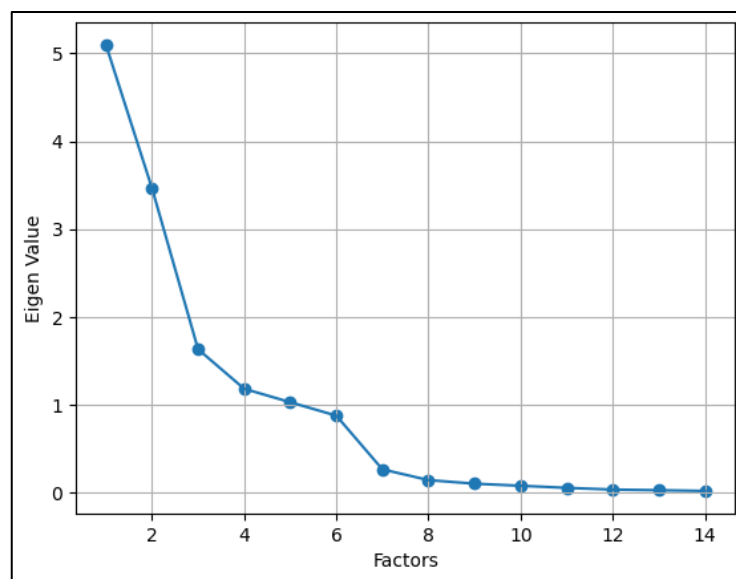


Figure 5-3- Scree plot for the Historical dataset

It is a little bit difficult to choose the number of factors by identifying the elbow point of the scree plot. Instead, the Kaiser criterion can be applied to identify the number of factors that are needed. There are only five eigenvalues greater than one. Therefore, the chosen

number of factors is five. The following snapshot shows the factor loadings for each factor with the varimax rotation:

	Factor 1	Factor 2	Factor 3	Factor 4	Factor5
T1_Mat	0.067722	0.180762	0.960190	0.049464	0.188644
T1_AvgRunsScored	0.065009	0.870715	0.187589	0.187498	-0.017842
T1_AvgRunsConceded	-0.103996	0.857855	0.136097	-0.072318	0.157428
T1_AvgWktsTaken	0.162205	-0.116920	-0.009115	-0.048356	-0.112615
T1_AvgWktsLost	-0.179580	-0.438907	-0.041175	-0.774982	0.032698
T1_AvgBound	0.018929	0.981810	0.146783	0.130014	0.006910
T1_W/L	0.211832	-0.045676	0.130947	0.958456	-0.127862
T2_Mat	0.313609	0.298303	0.841770	0.143628	-0.040089
T2_AvgRunsScored	0.942821	0.026860	0.165152	0.155454	-0.037410
T2_AvgRunsConceded	-0.131551	0.085607	0.195575	-0.089184	0.942989
T2_AvgWktsTaken	0.643647	-0.029731	0.101953	0.154846	-0.557876
T2_AvgWktsLost	-0.829887	-0.010176	-0.060945	-0.178508	0.284378
T2_AvgBound	0.938401	0.061913	0.182413	0.111883	0.175883
T2_W/L	0.785743	-0.022922	0.038372	0.190670	-0.550373

Figure 5-4- Factor loading for the Historical dataset

- Factor 1 has high loadings for T2_AvgRunsScored, T2_AvgWktsTaken, T2_AvgWktsLost, T2_AvgBound, T2_W/L, and T1_AvgWktsTaken
- Factor 2 has high loadings for T1_AvgRunsScored, T1_AvgRunsConceded, and T1_AvgBound
- Factor 3 has high loading for T1_Mat and T2_Mat
- Factor 4 has high loadings for T1_AvgWktsLost and T1_W/L
- Factor 5 has high loadings only for T2_AvgRunsConceded

The variables in the same factor have high correlations with other variables in that factor. From this, we can group highly correlated variables together.

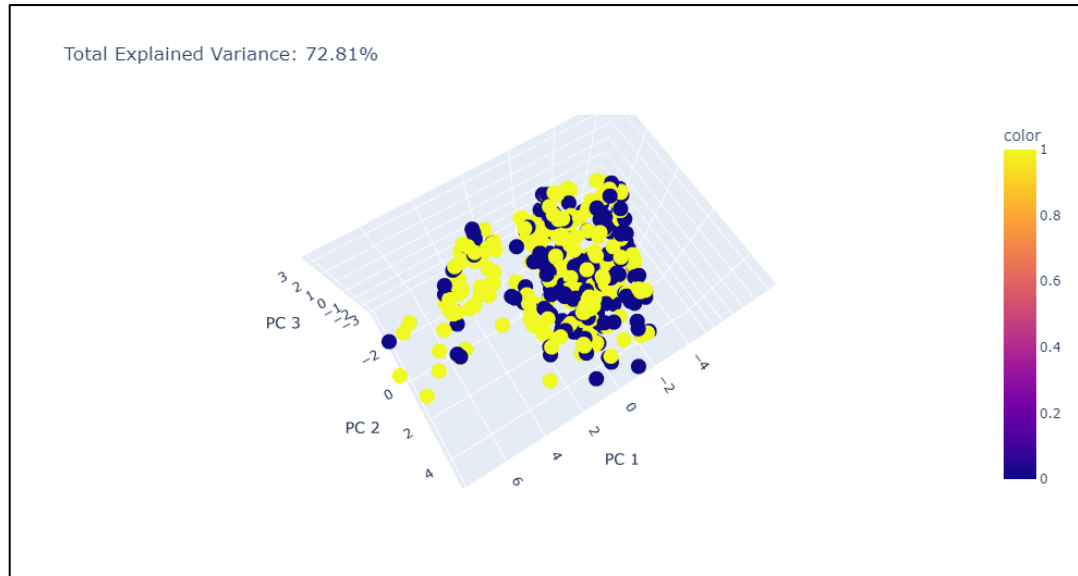


Figure 5-5- PCA plot for the Historical dataset

The above 3-D plot shows the first three components of the PCA. These three components explain 72.81% of the total variance. As mentioned in Chapter 3, the goal of creating this type of plot is to determine whether the data are linearly separable or not and, as a result, to determine which machine learning algorithms are appropriate for this dataset. The plot indicates that the data cannot be separated linearly. That suggests that nonlinear algorithms may give good results when building models.

5.2 Data Analysis – Twitter dataset

The below correlation plot reveals that there is a perfect negative correlation between Team 1 Twitter volume (T1_TwitterVol) and Team 2 Twitter volume (T2_TwitterVol). Another perfect negative correlation exists between T2 fans' sentiment score (T2_FansSent) and T1 Twitter volume (T1_TwitterVol). The correlation between the T2 Twitter volume (T2_TwitterVol) and the T2 fans' sentiment score (T2_FansSent) is a perfect positive one. In addition, there are other strong correlations that can be identified here. This indicates that multicollinearity exists in the dataset.

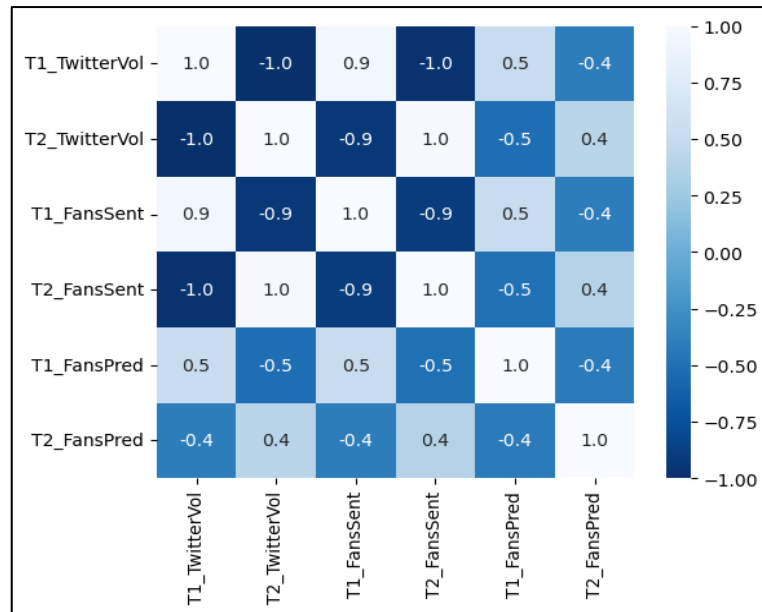


Figure 5-6- -Correlation plot for the Twitter dataset

Due to the few variables in the Twitter dataset, factor analysis was not conducted to determine the highly correlated variables group-wise.

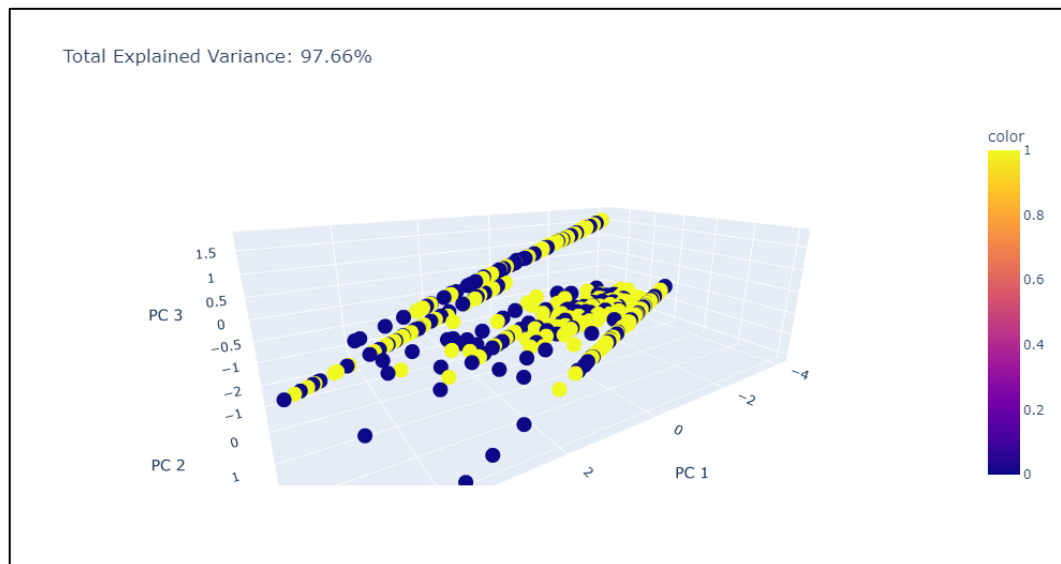


Figure 5-7- PCA plot for the Twitter dataset

According to Figure 5-7, for the Twitter dataset, the first three components explain 97.66% of the total variance. It seems that for this dataset as well, a non-linear machine learning

algorithm would be more suitable because the plot isn't able to show any linear separation of data.

5.3 Data Analysis – Historical + Twitter dataset

The same data analysis techniques were used for the "Historical+Twitter" dataset as they were for the "Historical" dataset. The below correlation-association plot, which is for the "Historical+Twitter" dataset, also suggests that multicollinearity exists in the dataset.

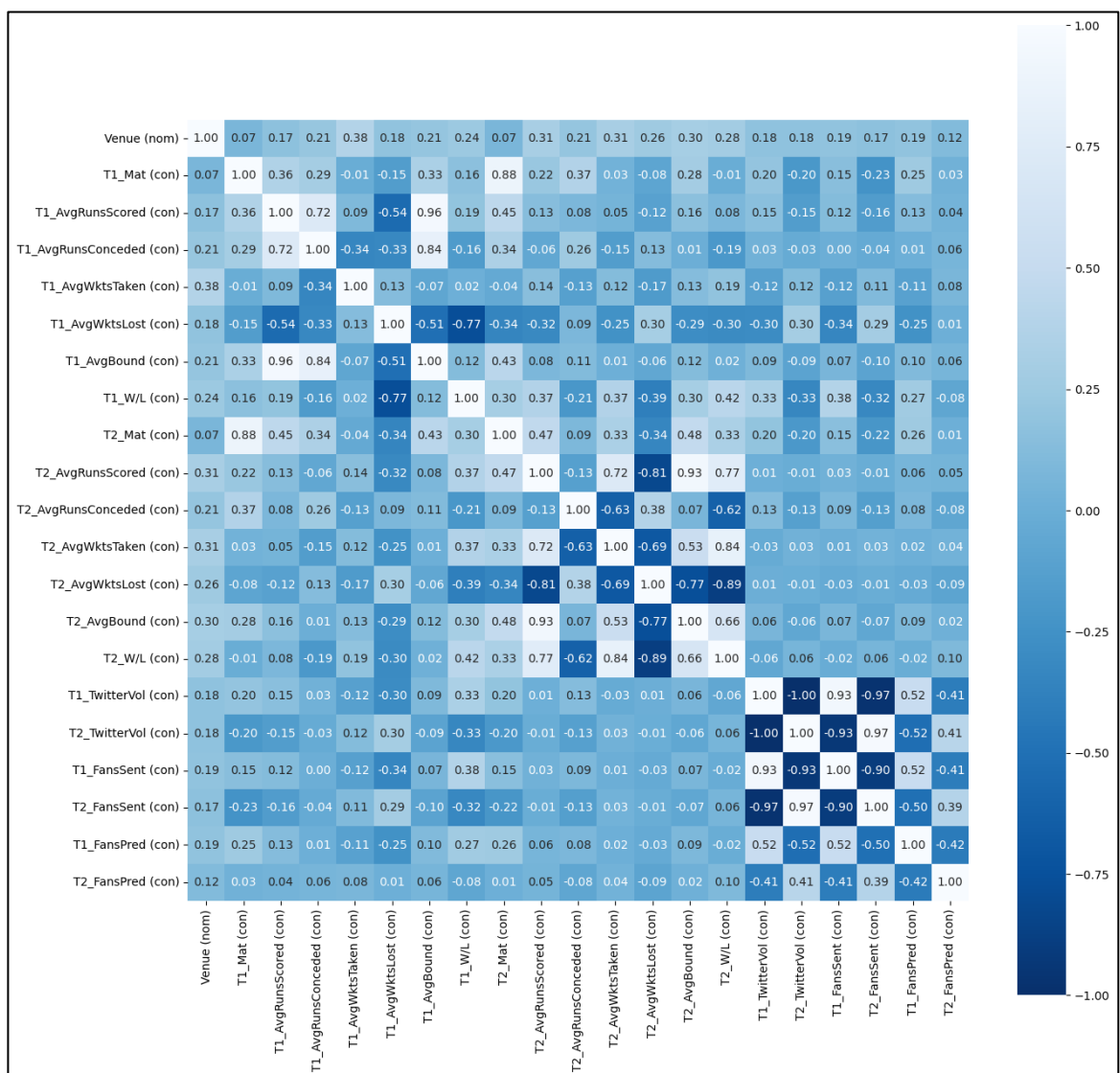


Figure 5-8- Correlation-association plot for the Historical+Twitter dataset

Both KMO and Bartlett satisfied the conditions by giving 0 for the p-value and 0.73 for KMO. The scree plot was then plotted in order to identify the number of factors. As in the "Historical" dataset, using the Kaiser criteria, the number of factors were identified.

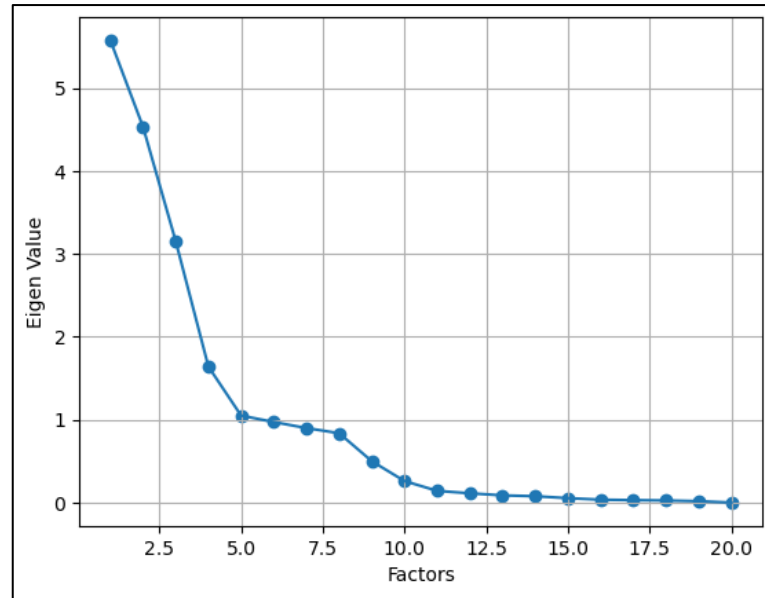


Figure 5-9- Scree plot for the Historical+Twitter dataset

There are only five eigenvalues bigger than one, as shown in the Figure 5-9. As a result, only five factors were included in the subsequent investigation. The snapshot below shows the loadings for each factor with varimax rotation:

	Factor 1	Factor 2	Factor 3	Factor 4	Factor5
T1_Mat	0.133472	0.086500	0.195092	0.949978	0.181993
T1_AvgRunsScored	0.068425	0.077978	0.898233	0.172374	-0.034702
T1_AvgRunsConceded	-0.060021	-0.132262	0.812640	0.137420	0.210030
T1_AvgWktsTaken	-0.119487	0.157226	-0.121247	-0.005516	-0.087062
T1_AvgWktsLost	-0.315665	-0.291338	-0.549740	-0.040194	0.245766
T1_Bound	0.005334	0.020336	0.991433	0.137599	0.012453
T1_W/L	0.377455	0.367164	0.156600	0.085642	-0.349297
T2_Mat	0.129381	0.340459	0.316553	0.828735	-0.048055
T2_AvgRunsScored	0.003412	0.956157	0.046780	0.149553	-0.019958
T2_AvgRunsConceded	0.110278	-0.172703	0.095011	0.192471	0.842071
T2_AvgWktsTaken	-0.027870	0.665404	-0.022124	0.090998	-0.541183
T2_AvgWktsLost	0.008510	-0.848418	-0.030478	-0.048838	0.277739
T2_Bound	0.050056	0.956885	0.075268	0.155051	0.221897
T2_W/L	-0.057901	0.810115	-0.005779	0.031075	-0.545277
T1_TwitterVol	0.980266	-0.004461	0.072964	0.055231	0.023774
T2_TwitterVol	-0.980266	0.004459	-0.072940	-0.055283	-0.023793
T1_FansSent	0.944015	0.027333	0.067024	0.002266	-0.019602
T2_FansSent	-0.947306	0.007908	-0.074345	-0.090589	-0.027706
T1_FansPred	0.551172	0.026213	0.063872	0.171062	-0.010962
T2_FansPred	-0.444093	0.053961	0.072483	0.040726	-0.049255

Figure 5-10- Factor loadings for the Historical+Twitter dataset

- Factor 1 has high loadings for T1_TwitterVol, T2_TwitterVol, T1_FansSent, T2_FansSent, T1_FansPred, T2_FansPred, and T1_W/L
- Factor 2 has high loadings for T2_AvgRunsScored, T2_AvgBound, T2_AvgWktsLost, T2_W/L, T2_AvgWktsTaken, and T1_AvgWktsTaken
- Factor 3 has high loading for T1_AvgRunsScored, T1_AvgRunsConceded, T1_AvgWktsLost, and T1_AvgBound
- Factor 4 has high loadings for T1_Mat and T2_Mat
- Factor 5 has high loadings only for T2_AvgRunsConceded

As mentioned previously, the variables are highly correlated with each other in that factor.

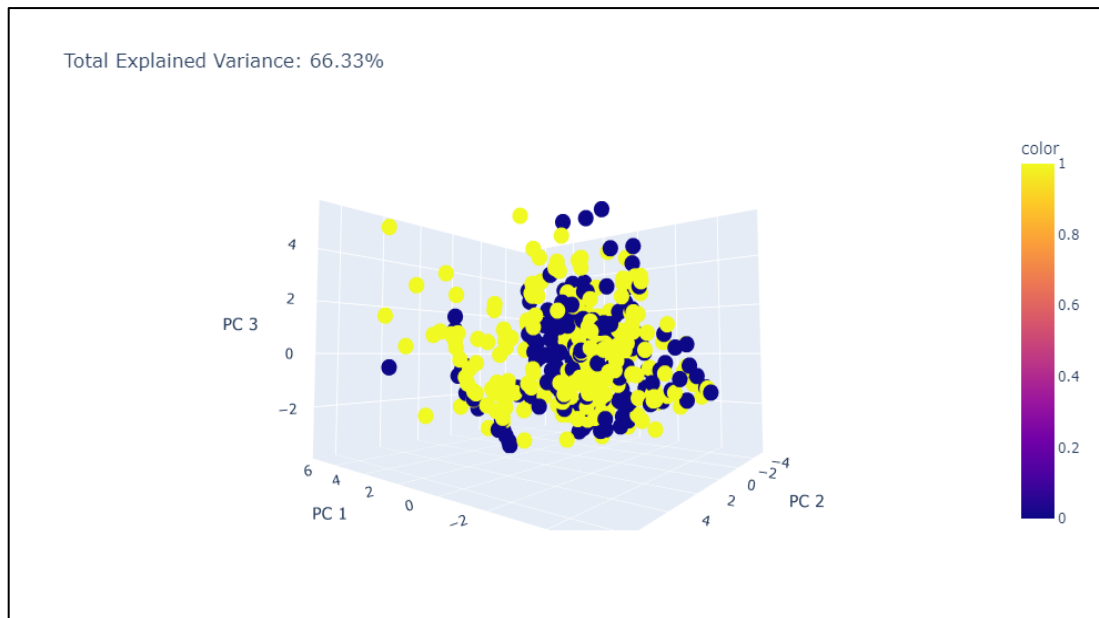


Figure 5-11 -PCA plot for the Historical+Twitter dataset

The PCA plot drawn for this dataset, Figure 5-11, indicates that data cannot be separable linearly since there is no pattern here either. Therefore, for this dataset as well, nonlinear algorithms may be appropriate when building models.

5.4 Conclusion

Since the datasets in this study were unbalanced, it is a good idea to take resampling approaches into account while fitting models. Multicollinearity was seen in all three datasets, namely "Historical," "Twitter," and "Historical+Twitter." As there was no linear separation of the data in PCA plots for any of the three datasets, non-linear machine learning algorithms may produce better results than linear algorithms. However, this cannot be guaranteed, but it is wise to pay more attention to non-linear algorithms when fitting models.

Chapter 6

Implementation and Results

This chapter discusses the research implementation as well as the main findings, including the sentiment analysis results. The chapter concludes by comparing the performance of the final predictive models and selecting the best model for future work.

6.1 Environments and software packages

Anaconda: Anaconda provides a combination of Python and R programming languages that are used for the advancement of scientific computing, machine learning, and deep learning. It includes a web-based interactive development environment (IDE) called Jupyter Notebook, which supports many languages, including Python. This study was performed using the Python programming language, and most of the codes were written and executed in Anaconda's Jupyter notebooks. The installed version of conda on the machine was 22.11.1.

Google Collaboratory (Google Colab): a free cloud-based service offered by Google that utilizes the Jupyter notebook environment, which is comparable to Anaconda. The crucial aspect is that it creates a virtual machine with specialized hardware, such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units), to expedite the task. Not only that, but it also offers very useful and powerful machine-learning libraries such as TensorFlow and PyTorch, and those are pre-installed. This study used Google Colab with the run type "GPU" rather than CPU (Central Processing Unit) for more complex and computationally demanding tasks.

Visual Studio Code (VS Code): a Microsoft-created open-source, free code editor with a wide range of extensions available for installation, including Python. The Streamlit application was written and executed using VS Code.

In addition, the following libraries were frequently utilized in this research:

- Numpy: for carrying out operations in mathematics and statistics
- Pandas: for effectively managing and storing datasets
- Matplotlib: for making charts and graphs to visualize data

- Scikit-learn (Sklearn): for preparing data, choosing models, choosing features, and evaluating models.
- NLTK (Natural Language Toolkit): for NLP tasks like tokenization and lemmatization

6.2 Data preparation

6.2.1 Data annotation

Due to a lack of resources and time constraints, only 3000 tweets were randomly selected from the collection, and those tweets were labeled to perform and evaluate the sentiment analysis. For this job, three annotators were employed, and they were each required to label the whole dataset with 1 if a tweet was positive and 0 if a tweet was negative. Then, the final labels were assigned based on the majority vote. The objective of following such a procedure was to reduce subjectivity and human bias to some extent.

6.2.2 Sentiment analysis

To create the two variables T1_FansSent (Team1 aggregated fans' sentiment score) and T2_FansSent (Team2 aggregated fans' sentiment score), identifying the sentiment of each tweet, whether positive or negative, was needed. For that, sentiment analysis was carried out.

As mentioned in Chapter 3, a Roberta-based model was mainly considered, and its performance was compared with VADER, a rule-based approach, and LSTM, a statistical-based approach. 3000 labeled tweets were split into three sets called training (70%), validation (15%), and testing (15%). Out of 2100 tweets in the training set, 1675 were positive. There were only 425 negative tweets, which indicates the dataset was imbalanced.

Twitter-roBERTa-base for Sentiment Analysis - UPDATED (2022)

To fine-tune the selected model on the dataset, the PyTorch Trainer was utilized. This process was done on Google Colab. Colab already had PyTorch installed, so all that was required to utilize it was to import the library. However, the transformer library had to be installed at the start using the pip command. Before fine-tuning the model, the three datasets were loaded, and the map method was used to apply the data preprocessing steps to the three datasets. Under the preprocessing step, a tokenizer called "AutoTokenizer" was used for the padding and truncating. The chosen model, "twitter-roberta-base-sentiment-latest," was then loaded from the Hugging Face model hub.

(<https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest>)

A `TrainingArguments` class that contains all the parameters to be tuned was defined, and a `Trainer` object was created with the model, training arguments, training, and validation datasets. The `train()` function was then used to tune the model. After completing all of these stages, the model's performance on the testing set was assessed. For later usage, the model was uploaded to the hub (*Cric-tweets-sentiment-analysis*, 2022). This model can be accessed by using this link: <https://huggingface.co/sppm/cric-tweets-sentiment-analysis>.

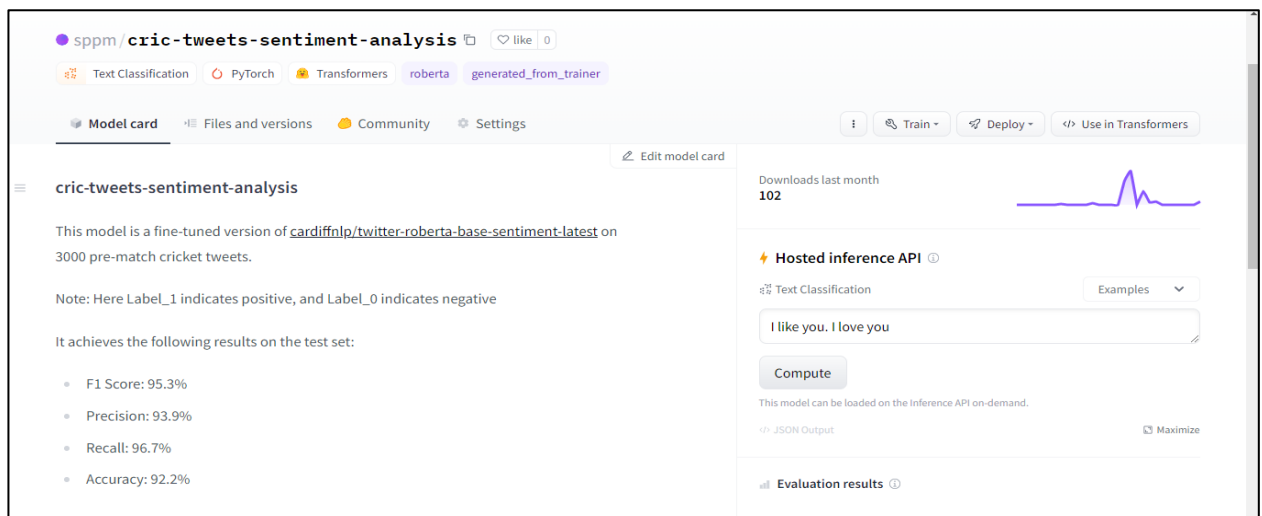


Figure 6-1 -Snapshot showing the fine-tuned model uploaded to the model hub

Following are the hyperparameters used in the model training:

Table 6-1- Best hyperparameters of the RoBERTa-based model

Hyperparameter	Value
learning_rate	5e-05
train_batch_size	16
eval_batch_size	16
seed	223
optimizer	Adam
num_epochs	200

And, the following results were obtained for the test set:

Table 6-2- Metric scores of the RoBERTa-based model for the test set

Metric	Value
ROC-AUC	84.2%
Precision	93.9%
Recall	96.7%
Accuracy	92.2%
F1 score	95.3%

All the metrics are shown as percentage values to make them more intuitive. The F1 score was used as the main metric in this study to evaluate and compare model performance. For this model, the F1 score of the training set was 97.4%, and for the validation set, it was 95.3%. These results indicated that the model is not overfitted. The class-wise F1 scores of this model were 95.3% for the positive class and 76.8% for the negative class, respectively.

VADER

Just the testing set was utilized to implement this strategy and assess the model's performance on that set because here VADER lacks a training phase. After downloading the VADER lexicon from NLTK, the sentiment analyzer was initialized. Next, based on the compound score, predictions were made. Tweets were classified as positive or negative depending on whether their compound score was greater than or equal to zero. When the score was greater than or equal to zero, tweets were labeled as positive; otherwise, they were labeled as negative. Lastly, the predictions were compared to the ground truth labels provided by annotators. The number of True Positives, True Negatives, False Positives, and False Negatives was used to compute the values for metrics, and the calculated values were as follows:

Table 6-3- Metric scores of VADER model for the test set

Metric	Value
Precision	90.7%
Recall	95.7%
Accuracy	88.4%
F1 score	93.1%

LSTM

TensorFlow, a pre-installed library in Google Colab, was used, and the GloVe embedding model from Stanford was downloaded to create this model. Then the training, validation, and test sets were cleaned using the "neattext" library for the task. It is not pre-installed in Colab; therefore, it was installed using the pip command before use. Utilizing this library, hashtags, user handles, multiple whitespaces, URLs, and punctuation were taken out of the tweets. Then the tokenization, lemmatization, and word embedding steps were performed. Tokenization and lemmatization steps were carried out by the NLTK library. For tokenization, NLTK's RegexpTokenizer was used, and for lemmatization, WordNetLemmatizer was utilized. Only the words from the 400,000-word GloVe embedding list were retained after all of these steps. Here, the 50-dimensional GloVe model was selected for the study. That means each word was represented by a 50-dimensional vector. Simple data analysis was performed on the training set to identify the maximum number of tokens in a tweet.

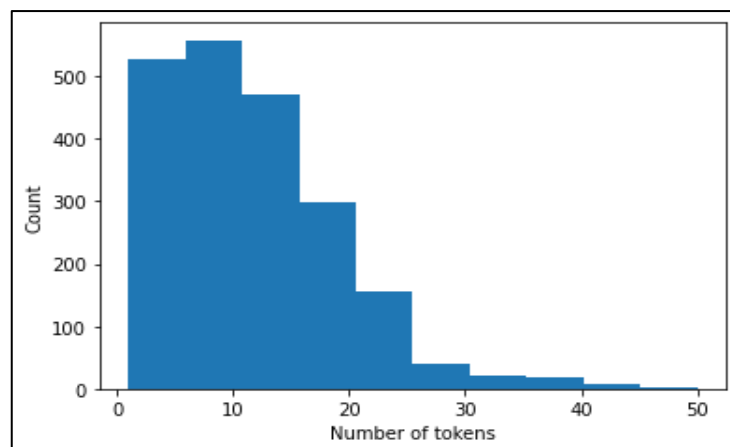


Figure 6-2- Histogram of the number of tokens in tweets

The plot shows that most of the tweets have between 0–20 tokens. Analysis gave the result for the maximum number of tokens, and that value was 50. The plot shows that value also. The length of each tweet was equalized by taking that value into account. If a tweet doesn't have the defined number of tokens, that was padded with zeros. Then the architecture of the LSTM model was defined, and the model was trained on the training set using suitable hyperparameters. Following are the used hyperparameters and their corresponding values:

Table 6-4- Best hyperparameters of the LSTM model

Hyperparameter	Value
Input_shape	(50,50)
return_sequences	True
units	First LSTM layer – 128 units The subsequent two layers – 64 units
dropout	0.2
activation	Sigmoid function
epochs	200
optimizer	Adam
loss	BinaryCrossEntropy
learning_rate	1e-04

Since the dataset is imbalanced, weights were added considering the class frequencies. An immediate evaluation was done on the validation set, and the final performances were done on the test set. Following are the results for the test set that were obtained from the LSTM model:

Table 6-5-Metric scores of the LSTM model for the test set

Metric	Value
ROC-AUC	75.8%
Precision	92.2%
Recall	83.7%
Accuracy	80.9%
F1 score	87.8%

The LSTM model's F1 score for the training set was 92.2%, while for the validation set, it was 89.6%. These values can be raised by utilizing a large training set, however, that was not feasible due to time and resource constraints. For the positive class and the negative class, the class-wise F1 scores of this model were 88% and 56%, respectively. Since the negative class had a very low F1 score even after adding weights, this model was not as good as the fine-tuned RoBERTa model.

The optimal technique for sentiment analysis was then found by comparing the test-set F1 scores of these three models. Based on these results, the fine-tuned RoBERTa model worked better than the other two models, as expected. Finally, the RoBERTa model was applied to determine the sentiment of the entire collection of tweets.

Table 6-6 Final results comparison of sentiment analysis

Fine-Tuned Roberta	VADER	LSTM
95.3%	93.1%	87.8%

For both RoBERTa and LSTM models, early stopping was used in order to avoid overfitting.

6.2.3 Preprocessing data

After formulating the three datasets, Historical, Twitter, and Historical+Twitter, as discussed in Chapter 4, categorical variables were encoded using one-hot encoding, and all of the variables were standardized using StandardScaler(). The dataset was unbalanced, as previously noted. Hence, SMOTE was then applied to each dataset.

6.3 Performance of the final models

PCA plots for all three datasets, "Historical," "Twitter," and "Historical+Twitter", showed that non-linear classification algorithms were suitable. But could not be entirely assured on that claim, as was discussed in Chapter 4. And hence, both linear and non-linear algorithms were used, but the non-linear methods received more focus. Each dataset was exposed to the machine learning algorithms chosen based on earlier research and the analysis of this study. To evaluate the models, three repeats of 10-fold cross-validation were used in this study.

6.3.1 Historical data model

The findings for the test set of the Historical dataset were as follows:

Table 6-7-Primary results of the Historical data model

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Logistic Regression	56.0%	60.0%	54.5%	56.7%	57.1%
SVM	60.9%	65.2%	54.5%	60.6%	59.4%
KNN	58.1%	62.2%	50.9%	57.7%	56.0%
Naïve Bayes	58.4%	64.1%	45.5%	57.7%	53.2%
Random Forest	61.9%	66.0%	56.4%	61.5%	60.8%
XGBoost	60.8%	64.6%	56.4%	60.6%	60.2%

Note: In this study, for all three datasets, when building models using logistic regression, primarily three models were trained based on the penalty: logistic with ridge penalty, logistic with lasso penalty, and logistic with the elastic net penalty. Out of these three models, the one that gave the best results is included in this paper under the logistic regression model along with its metric values.

In order to compare the models' performances, mainly the F1 score was taken into consideration. This table shows that Random Forest and XGBoost both provided good results compared to the other classifiers, with Random Forest having slightly higher results than XGBoost. In order to determine which variables were most important, respective importance plots were created.

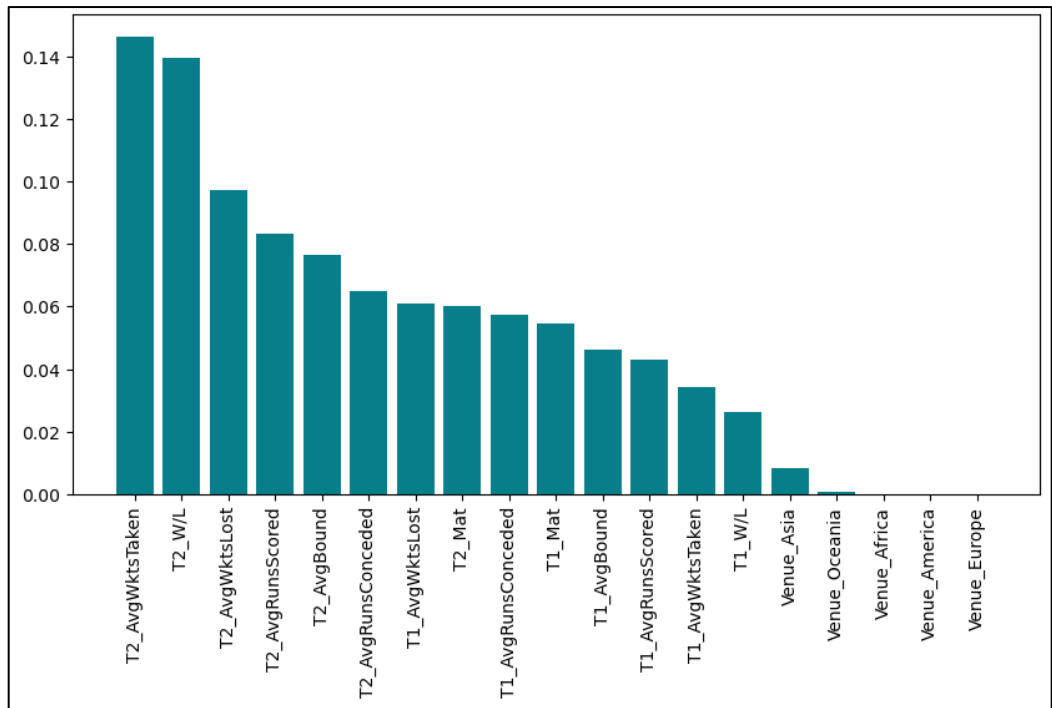


Figure 6-3- Feature importance plot of Random Forest model for the Historical dataset

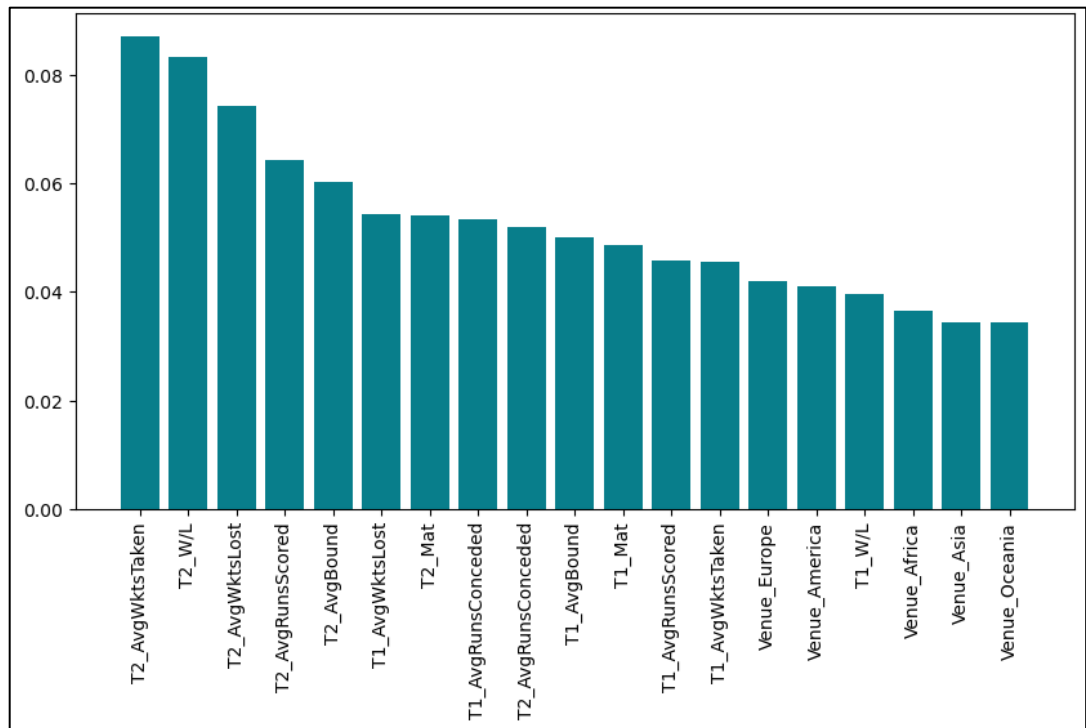


Figure 6-4- Feature importance plot of XGBoost for the Historical dataset

According to Figure 6-3 and Figure 6-4, "Venue" is one of the least important variables. After removing various sets of unimportant variables, the best results for both models were obtained by removing only the "Venue" variable. With this step, the best results came from XGBoost, with a 61.7% F1 Score, once the "Venue" variable was removed, and this model outperformed all the other models built up to now for the "Historical" dataset in this study.

Table 6-8-Results of the Historical data model without 'Venue'

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Random Forest without Venue	60.7%	64.0%	58.2%	60.6%	61.0%
XGBoost without Venue	60.6%	63.5%	60.0%	60.6%	61.7%

The feature importance plot of the XGBoost model without Venue variable was then presented as in Figure 6-5.

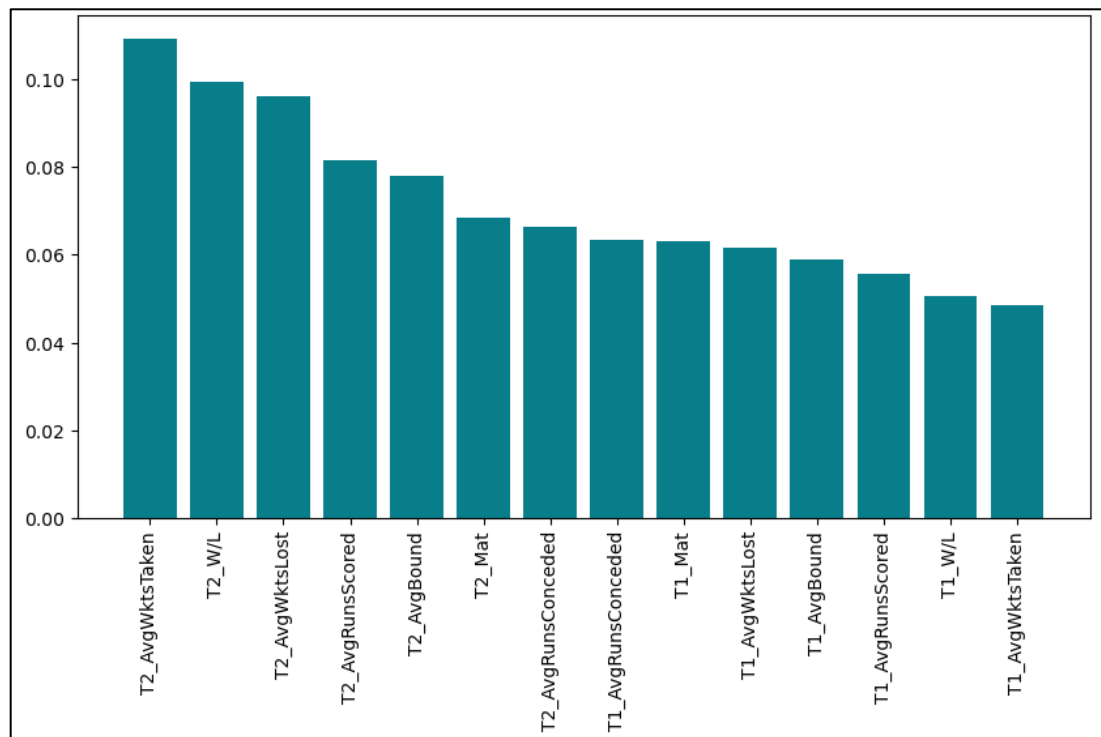


Figure 6-5- Feature importance plot of XGBoost without 'Venue' for the Historical dataset

The dataset's multicollinearity was revealed by data analysis in the previous chapter. The next step was to see if the F1 score of this model could be further increased after reducing multicollinearity. To do this, factor analysis was performed to group highly correlated variables. The findings from the second method, variable reduction, outperformed the first approach, the factor score technique, discussed in Chapter 3. Hence, only the outcomes of the variable reduction method are shown here. Results from the first method are shown in Appendix C.

Since the variables in the same factor are highly correlated, the variables that are most important from each factor were retained after going through the variable importance plot. According to the factor analysis results that were shown in the data analysis chapter, variables were grouped under each factor based on high loadings as follows:

Table 6-9- Correlated variables under each factor for the Historical dataset

Factor1	Factor2	Factor3	Factor4	Factor5
T2_AvgRunsScore	T1_AvgBound	T1_Mat	T1_W/L	T2_AvgRunsConceded
T2_AvgBound	T1_AvgRunsScore	T2_Mat	T1_AvgWktsLost	
T2_AvgWktsLost	T1_AvgRunsConceded			
T2_W/L				
T2_AvgWktsTaken				
T1_AvgWktsTaken				

The most important variable from each factor was chosen as input for the new set of models. The XGBoost without Venue model's variable importance plot was used to find those variables. That means the best results were achieved by limiting the analysis to T2_AvgWktsTaken, T1_AvgRunsConceded, T2_Mat, T1_AvgWktsLost, and T2_AvgRunsConceded variables. The following revised correlation plot, Figure 6-6, demonstrates that multicollinearity is also reduced to an extent:

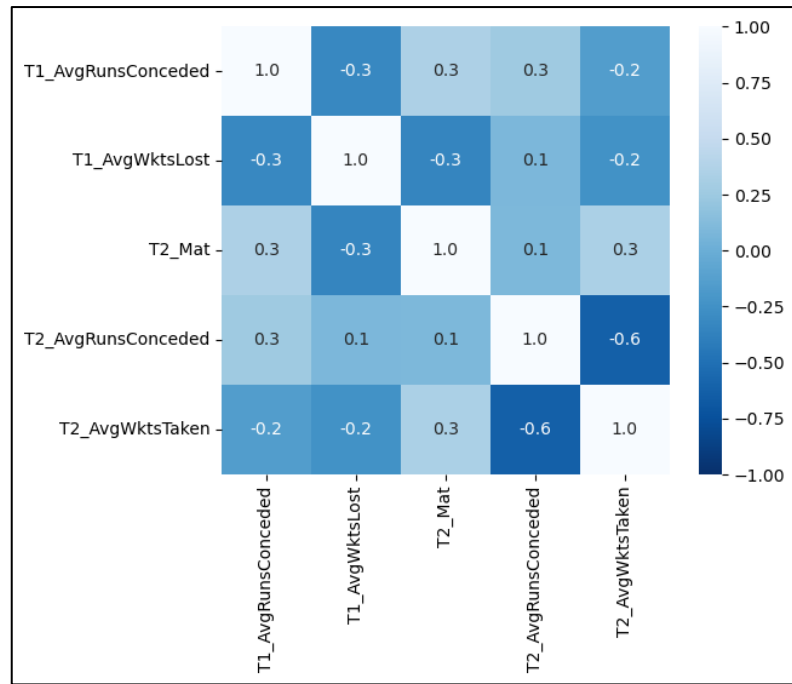


Figure 6-6- Correlation plot after variable reduction for the Historical dataset

The outcomes for the test set from the machine learning classifiers that were considered are displayed in Table 6-10.

Table 6-10- Results of the Historical model after variable reduction

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Logistic Regression	59.7%	62.7%	58.2%	59.6%	60.4%
SVM	56.4%	58.6%	61.8%	56.7%	60.2%
KNN	60.9%	65.2%	54.5%	60.6%	59.4%
Naïve Bayes	57.0%	60.4%	52.7%	56.7%	56.3%
Random Forest	60.5%	63.0%	61.8%	60.6%	62.4%
XGBoost	61.1%	62.3%	69.1%	61.5%	65.5%

The results obtained from this method were better than the results of other previous models. Then it can be concluded that XGBoost using the variable reduction method combined with factor analysis is the best model to predict the match outcome using only historical data. With a 65.5% F1 score, this model can predict the outcome of a T20 international cricket

match. It accurately predicted 66% of Team 1's and 57% of Team 2's victories. The training set F1 score was 65.9%, indicating the model was not overfit.

Before training this XGBoost model, the grid search gave the following set of hyperparameter values as the optimal set, and the model was trained based on these hyperparameters.

Table 6-11- Best hyperparameters of the best model for the Historical dataset

Hyperparameter	Value
learning_rate	1e-04
n_estimators	2000
max_depth	6
min_child_weight	6
subsample	0.25
colsample_bytree	0.4

6.3.2 Twitter data model

The performance measures of each classifier used in the study for all the variables in the Twitter dataset were as follows:

Table 6-12- Primary results for the Twitter data model

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Logistic Regression	60.5%	63.0%	61.8%	60.6%	62.4%
SVM	59.1 %	60.7 %	67.3 %	59.6 %	63.8 %
KNN	60.4%	62.5%	63.6%	60.6%	63.1%
Naïve Bayes	59.0%	60.3%	69.1%	59.6%	64.4%
Random Forest	58.9 %	60.0 %	70.9 %	59.6 %	65.0 %
XGBoost	61.0 %	61.9 %	70.9 %	61.5 %	66.1 %

This table shows that XGBoost and Random Forest produced the best outcomes, but XGBoost performs slightly better than Random Forest. Feature importance plots for XGBoost and Random Forest were generated to examine whether the model's performance could be boosted by removing less important variables. The following are the feature importance plots for the two models:

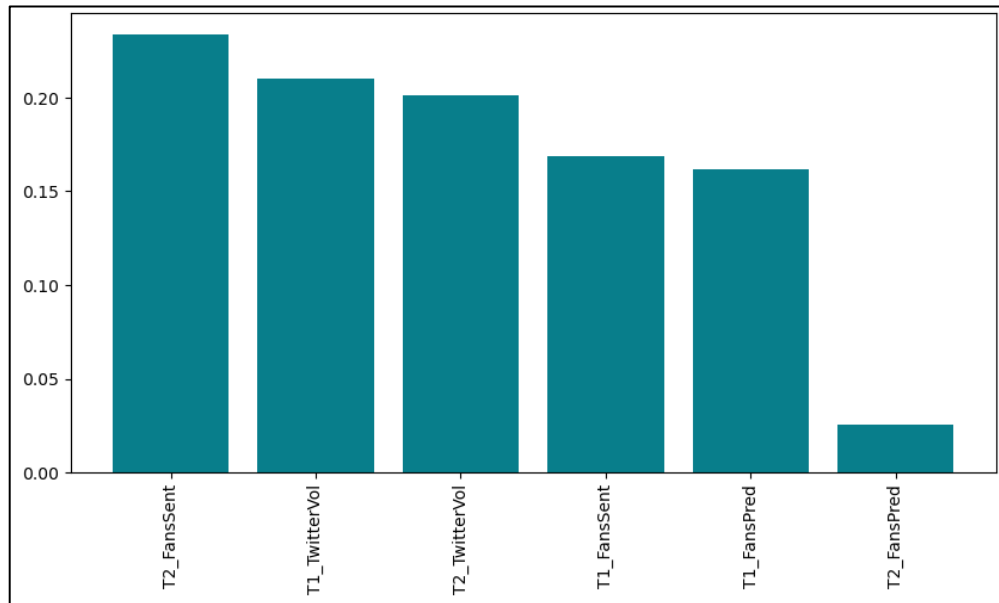


Figure 6-7- Feature importance plot of Random Forest for the Twitter dataset

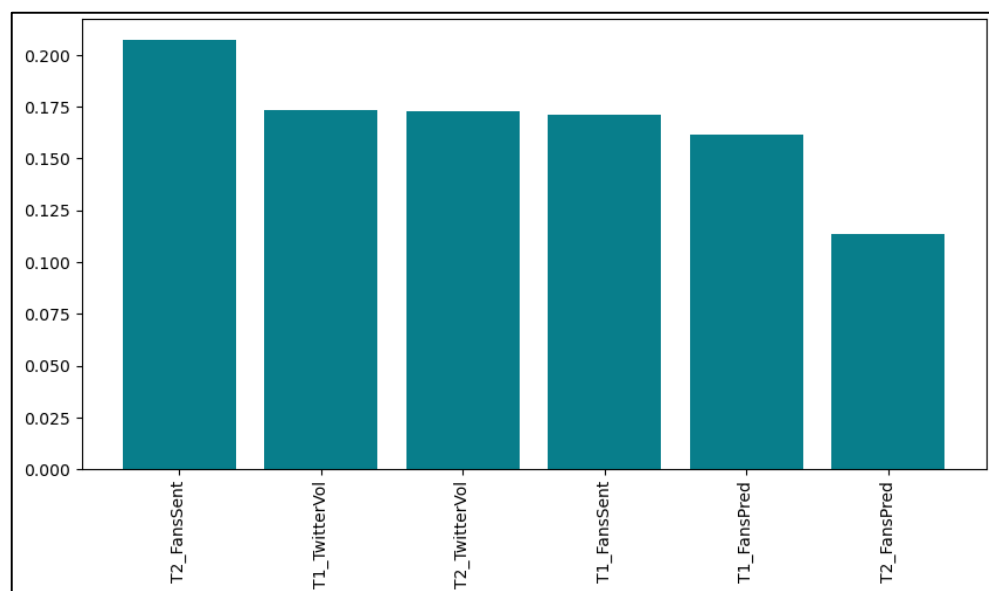


Figure 6-8- Feature importance plot of XGBoost for the Twitter dataset

The fans' prediction variables for the two teams appear to be the least important variables in both plots. To get the best outcomes, a variety of potential combinations of unimportant variables were tested. However, eliminating both the T1_FansPred and T2_FansPred variables led to the best performance.

Table 6-13- Results of the Twitter data model without fans' predictions

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Random Forest without T1_FansPred & T2_FansPred	59.8 %	60.6 %	72.7 %	60.6 %	66.1 %
XGBoost without T1_FansPred & T2_FansPred	65.5%	64.7%	80.0%	66.3%	71.5%

According to Table 6-13, after removing the T1 FansPred and T2 FansPred variables, the XGBoost algorithm produced the best outcomes. The minimal number of variables in this dataset restricted the use of the factor analysis approach, as was described in the data analysis chapter. As a result, the "XGBoost without T1 FansPred & T2 FansPred" model was chosen as the top model for predicting the result of the match using just features gathered from Twitter. The training F1 score for the model was 74.7%, and its class-wise F1 scores were 72% and 59%, respectively, for Team 1 wins and Team 2 wins. Following are the hyperparameters that were used to train this best-fit model:

Table 6-14- Best hyperparameters of the best model for the Twitter dataset

Hyperparameter	Value
learning_rate	1e-04
n_estimators	2000
max_depth	4
min_child_weight	2
subsample	0.6
colsample_bytree	0.75

6.3.3 Historical+Twitter model

The same process used for generating the Historical model was applied to the Historical+Twitter dataset. Initially, models were created utilizing all the variables in the Historical+Twitter dataset, and the outcomes for each algorithm were as follows:

Table 6-15- Primary results for the Historical+Twitter data model

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Logistic Regression	68.1%	69.6%	70.9%	68.3%	70.3%
SVM	69.0 %	70.2 %	72.7 %	69.2 %	71.4 %
KNN	63.8%	68.1%	58.2%	63.5%	62.7%
Naïve Bayes	64.1%	65.5%	69.1%	64.4%	67.3%
Random Forest	65.5%	68.6%	63.6%	65.4%	66.0%
XGBoost	65.5%	68.6%	63.6%	65.4%	66.0%

Table 6-15 shows that the best F1 score is coming from SVM, and the important thing is that the parameter grid chose the linear kernel for this best-fit model. The next step was to create the variable importance plot for this model. However, SVM doesn't have a function to calculate feature importance like in Random Forest and XGBoost. Therefore, as discussed in Chapter 3, the magnitude of the feature coefficients was used to create the feature importance plot for this model.

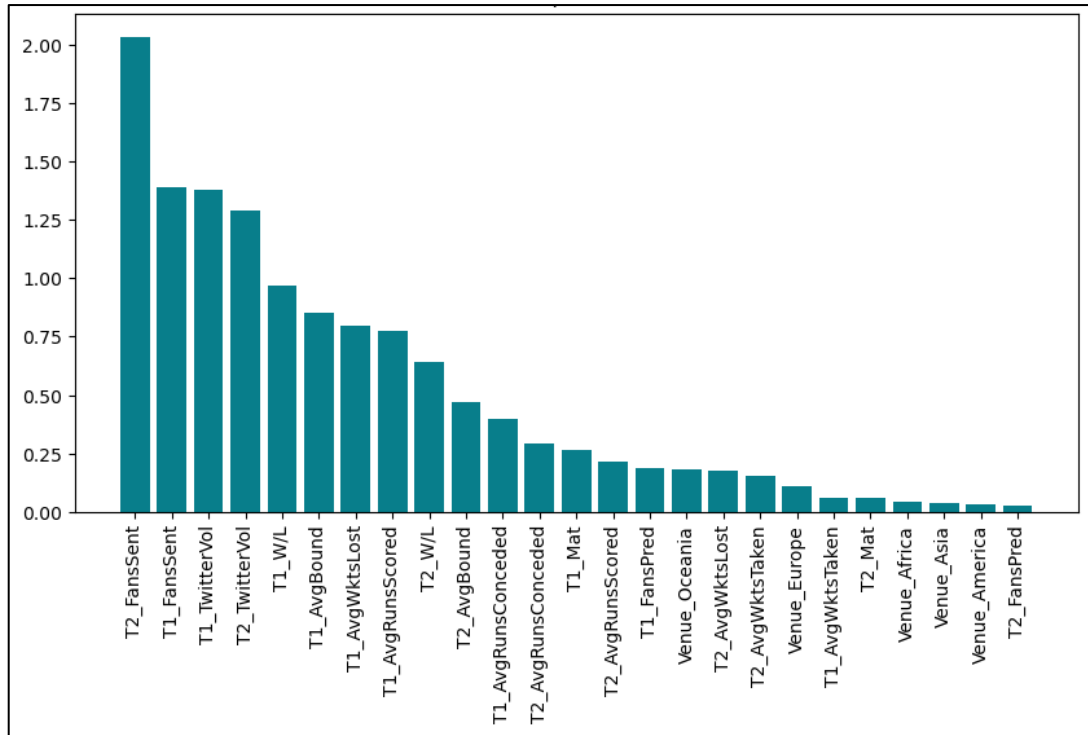


Figure 6-9- Feature importance plot of SVM for the Historical+Twitter dataset

According to Figure 6–9, T2_FansPred, T2_Mat, Venue, and T1_AvgWktsTaken are variables that are less important than the others. However, the F1 score was able to increase with the removal of two variables, T2_Mat and T2_FansPred.

Table 6-16- Results of the Historical+Twitter data model without T2_Mat & T2_FansPred

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
SVM without T2_Mat & T2_FansPred	67.0%	67.8%	72.7%	67.3%	70.2%

But as shown in Table 6-16, this model did not perform better than the previous one, which was built by incorporating all variables. Therefore, considering the feature importance plot of the first model, important variables were selected from each factor.

Table 6-17- Correlated variables under each factor for the Historical+Twitter dataset

Factor1	Factor2	Factor3	Factor4	Factor5
T1_W/L	T1_AvgWktsTaken	T1_AvgRunsScored	T1_Mat	T2_AvgRunsConceded
T1_TwitterVol	T2_AvgRunsScored	T1_AvgRunsConceded	T2_Mat	
T2_TwitterVol	T2_AvgWktsTaken	T1_AvgWktsLost		
T1_FansSent	T2_AvgWktsLost	T1_AvgBound		
T2_FansSent	T2_AvgBound			
T1_FansPred	T2_W/L			
T2_FansPred				

As in the Historical model, the factor score method did not perform better than the variable reduction method. The findings of the variable reduction approach are explained here, while the results of the factor score method are shown in Appendix Table C-2.

After taking into account various potential combinations, the two most important variables from each of the first two factors were selected, and from the remaining factors, the most important variable from each factor was chosen. One of the least important variables, Venue, was not considered because the optimal results were given by T1_FansSent, T2_FansSent, T2_AvgBound, T2_W/L, T1_AvgBound, T1_Mat, and T2_AvgRunsConceded when model fitting. The following is the correlation plot obtained for the selected variables, and according to that, the multicollinearity was also reduced to some extent:

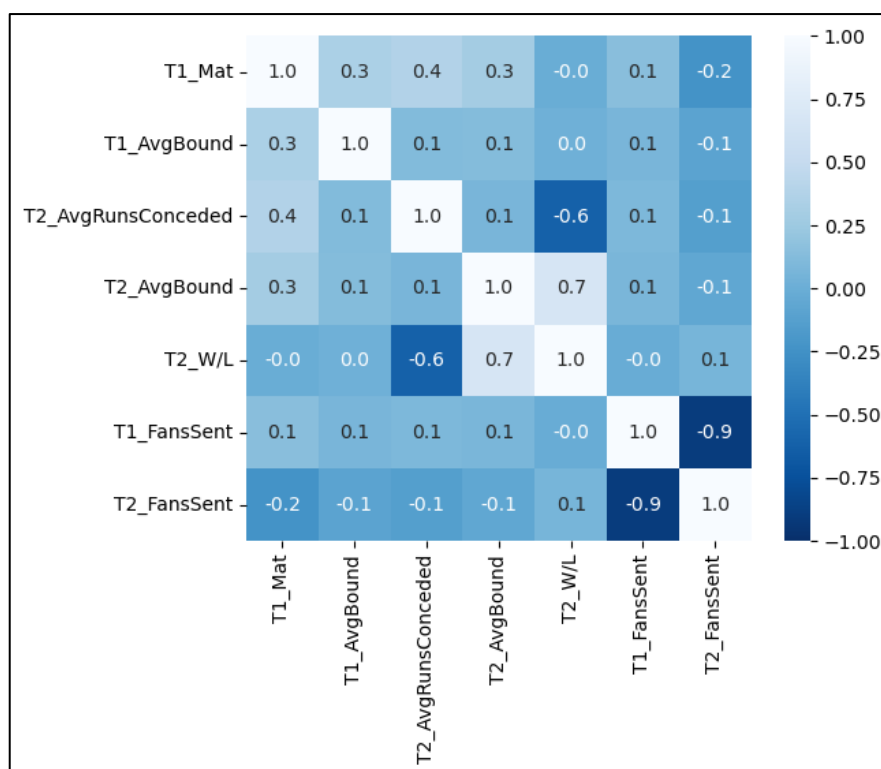


Figure 6-10-Correlation plot after variable reduction for the Historical+Twitter dataset

As shown in Table 6-18, XGBoost gave the best results for this dataset (a 73.7% F1 score for the test set), and the classification report showed an F1 Score of 74% and 68% per class, respectively, for Team 1 and Team 2. The F1 score for the training set was 76.8%. After getting these outcomes, a voting classifier was modeled using Logistic Regression, SVM, Random Forest, and XGBoost to determine whether or not the results could be improved, but there was no improvement.

Table 6-18- Results of the Historical+Twitter model after variable reduction

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Logistic Regression	64.2%	66.1%	67.3%	64.4%	66.7%
SVM	63.2%	64.9%	67.3%	63.5%	66.1%
KNN	60.5%	63.0%	61.8%	60.6%	62.4%
Naïve Bayes	63.3%	65.5%	65.5%	63.5%	65.5%
Random Forest	66.5%	70.0%	63.6%	66.3%	66.7%
XGBoost	70.8%	71.2%	76.4%	71.2%	73.7%
Voting classifier	67.2%	69.1%	69.1%	67.3%	69.1%

The optimal set of hyperparameters for the XGBoost model was as follows:

Table 6-19- Best hyperparameters of the best model for the Historical+Twitter dataset

Hyperparameter	Value
learning_rate	1e-04
n_estimators	2000
max_depth	7
min_child_weight	1
subsample	0.25
colsample_bytree	0.2

After fitting models for all three datasets, the best results of each model are summarized together with the variables and classifiers used to train those models as follows:

Table 6-20- Final results summarization

Model	Input Variables	Best classifier	F1 Score
Historical model	T2_AvgWktsTaken, T1_AvgRunsConceded, T2_Mat, T1_AvgWktsLost, T2_AvgRunsConceded	XGBoost	65.5%
Twitter model	T1_TwitterVol, T2_TwitterVol, T1_FansSent, T2_FansSent	XGBoost	71.5%
Historical+ Twitter model	T1_FansSent, T2_FansSent, T2_AvgBound, T2_W/L, T1_AvgBoud, T1_Mat, T2_AvgRunsConceded	XGBoost	73.7%

The above summary shows that the model, which used both Historical and Twitter data, gave the best results. The feature importance plot for this best model is as follows:

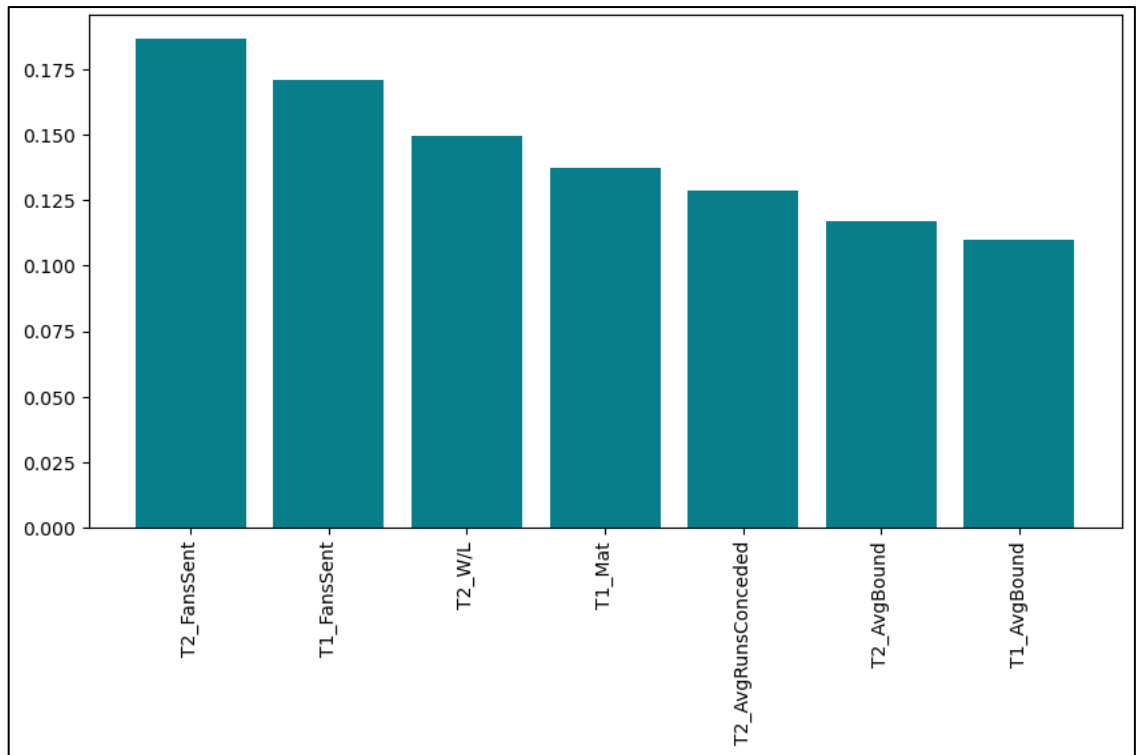


Figure 6-11- Feature importance plot of the final best model

According to the plot, the newly created two features using sentiment analysis are the most important variables. Next, to understand how these variables affect the model's predictions, partial dependency plots were created. Following are some of the partial dependence plots created for some variables:

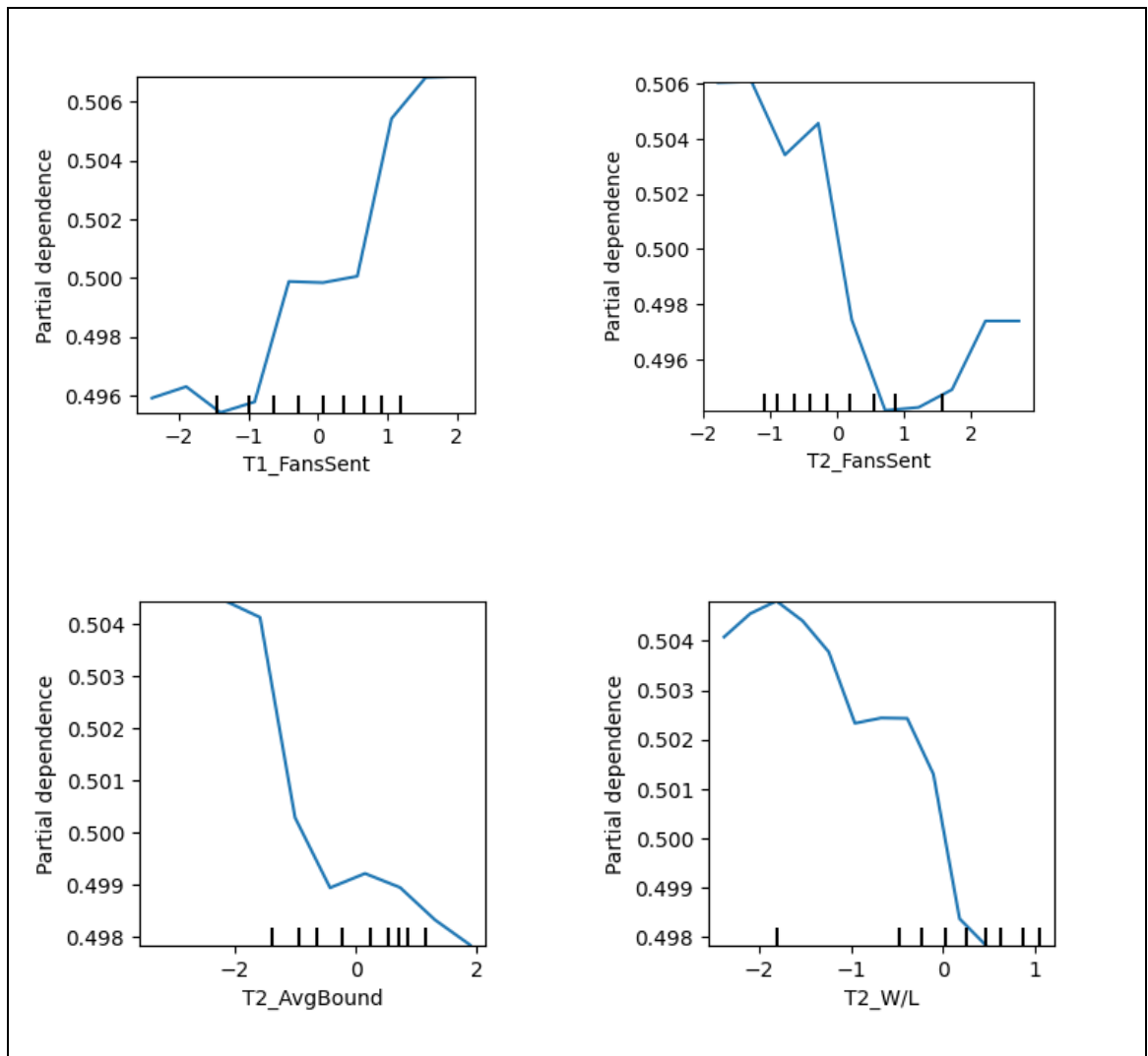


Figure 6-12- Partial dependency plots

Note that the X-axis of these graphs shows the standardized values.

According to the partial dependency plot of the T1_FansSent, when Team 1 has higher values for their sentiment score, they have a high chance of being the winner. The second partial dependency plot, which is for the Team 2 fans' sentiment score, shows that, up to nearly 0.9, if the T2_FansSent value increases, then the winning chance of Team 1 decreases. But after that, it begins to increase. When looking at the reason for this, it was noted that it was due to a lack of data. Therefore, it can be concluded that overall, when the T2_FansSent increases, the probability of Team 1 winning decreases. When considering the historical variables, there seems to be a negative association between T2_AvgBound and the winning probability of Team 1. Another negative association exists between T2_W/L and Team 1 winning chance. If team 2 had a higher win-loss ratio and took more boundaries per over, then team 1 would have a lower chance of winning.

6.4 Check the effectiveness of the proposed methodology

To check the effectiveness of the proposed methodology, predictions from the best model for the T20 World Cup 2022 matches were compared with the bookmakers' pre-match predictions for those matches. Since pre-match decimal odds for both teams were available for each match, the probability of a team winning a game can be easily calculated. Divide 1 by the decimal odds to achieve this. It indicates the team with lower decimal odds has a higher chance to win the match, according to the bookmakers. The following table contains the average pre-match betting odds obtained by oddsportal.com, bookmakers' predictions for each match, the proposed methodology's predictions, and the actual outcome:

Table 6-21- Data to check the effectiveness

Team1	Team2	Winner	Team1_ Decimal Odds	Team2_ Decimal Odds	Bookma kers' predictio n	proposed methodo logy's predictio ns
Australia	New Zealand	New Zealand	1.43	2.78	Australia (Team1)	New Zealand (Team2)
India	Pakistan	India	1.52	2.52	India (Team1)	India (Team1)
Australia	Sri Lanka	Australia	1.18	4.7	Australia (Team1)	Australia (Team1)
South Africa	Banglade sh	South Africa	1.2	4.49	South Africa (Team1)	South Africa (Team1)
New Zealand	Sri Lanka	New Zealand	1.36	3.12	New Zealand (Team1)	New Zealand (Team1)
India	South Africa	South Africa	1.62	2.28	India (Team1)	India (Team1)

England	New Zealand	England	1.65	2.2	England (Team1)	England (Team1)
India	Banglade sh	India	1.11	6.24	India (Team1)	India (Team1)
Pakistan	South Africa	Pakistan	2.27	1.62	South Africa (Team2)	South Africa (Team2)
England	Sri Lanka	England	1.22	4.17	England (Team1)	England (Team1)
Pakistan	Banglade sh	Pakistan	1.29	4.1	Pakistan (Team1)	Pakistan (Team1)
Pakistan	New Zealand	Pakistan	2.05	1.76	New Zealand (Team2)	Pakistan (Team1)
India	England	England	1.74	2.08	India (Team1)	India (Team1)
Pakistan	England	England	2.33	1.6	England (Team2)	England (Team2)

The green colour shows the correct predictions, and the red colour shows the wrong predictions.

Out of 14 matches, only nine were successfully predicted by bookmakers, and 11 matches were correctly predicted by the best model proposed in this study. However, rather than the number of matches that were accurately predicted, bookmakers make their predictions based on the odds (Ul Mustafa et al., 2017). Therefore, evaluating the two approaches by comparing the profit/loss amounts is better. Suppose \$1 was bet for each match; that means for all matches, a total of \$14 was placed for betting. Based on the bookmakers' predictions, only a total payout of \$12.13 could have been obtained, so the bettor has to face a loss. But a total payout of \$16.96 (a profit of \$2.96) could have been earned by following this proposed methodology. This implies that the model proposed by this study follows a different prediction approach than that of the bookmakers.

Chapter 7

Discussion and Conclusion

This chapter discusses the results and findings of the study, the challenges faced and actions taken to address them, the limitations of the study, and finally, suggestions for future work.

7.1 Discussion

As mentioned earlier, for sentiment analysis of cricket match tweets, most of the previous studies have considered rule-based or machine learning-based approaches. According to the literature review, A. N. Wickramasinghe & Yapa (2019) obtained a good sentiment analyzer compared to other studies, and their model achieved an accuracy of up to 85%. The fine-tuned RoBERTa model trained in this study had a high F1 score of 95.3% and an accuracy of 92.2%. Therefore, compared to the models built for sentiment detection in cricket tweets in previous studies, this fine-tuned RoBERTa model is a better model. This model is pre-trained on a large tweet set to specifically identify tweet sentiment. Therefore, that can be a reason for obtaining higher performance.

Predicting the match winner of T20I matches before starting the match using historical data and Twitter data was the main objective of this study. The following table shows the results obtained in this study for those predictive models:

Table 7-1- Predictive model comparison using F1 scores

Model	Using the complete dataset	After removing the least important variables	After using the variable reduction method
Historical	60.8% (Random Forest)	61.7% (XGBoost)	65.5% (XGBoost)
Twitter	66.1% (XGBoost)	71.5% (XGBoost)	-
Historical+Twitter	71.4% (SVM)	70.2% (SVM)	73.7% (XGBoost)

According to Table 7-1, the Historical model achieved its best F1 score (65.5%) after using the variable reduction technique combined with factor analysis. However, most of the past

studies have considered franchise-based league T20 historical match data or a combination of T20 international and league historical match data. Lamsal & Choudhary (2018) obtained a 72% F1 score for their historical data model, which was built to predict the IPL match winner 15 minutes before starting the match. That best performance came from the Multilayer Perceptron model, which is a type of neural network architecture. One of the factors contributing to their study's high F1 score can be the use of a deep learning technique like a neural network architecture.

When considering the models built using only Twitter data, Ul Mustafa et al. (2017) were able to obtain up to 75% correct predictions for their large dataset. But in this case, the model using solely Twitter data only managed to get 71.5% as the best F1 score. This can be due to the difference in datasets between the two studies. Twitter volume and fans' sentiment score variables were utilized in the same way as they were in the study by Ul Mustafa et al. (2017). The only difference in variables is that they considered fans' predicted total score as a variable, but in this study, fans' written predictions were taken into account when creating that feature due to the lack of tweets with that pattern. However, none of the final models in this study used this newly created feature. That indicates this feature doesn't add any value to the study.

According to the author's knowledge, no study up to now has built a model to predict the T20I match outcome before it starts by integrating both Historical and Twitter data. This study was able to achieve an F1 score of 73.7% for that model by surpassing both individual data models. It was noted that this best model, Historical+Twitter, could beat the predictions of the bookmakers and generate a profit.

7.2 Problems encountered and solutions

- It was difficult to run the models built using PyTorch and TensorFlow locally. Therefore, Google Colab was used with the GPU runtime to run these codes.
- In order to get particular values for each variable in the Twitter dataset, the same set of codes had to be executed for each team in each match, which was not easy. So, the app created to identify the sentiment of tweets was extended to generate values for all basic Twitter variables.
- While creating the list of hashtags, it was seen that other sports teams have also used some of the same hashtags used for cricket teams. For example, the

Bangladesh cricket team and the Detroit Tigers, a baseball team, use the hashtag #Tigers. Therefore, this can cause some problems. Due to that reason, after a careful look, these types of hashtags were removed from the list

7.3 Limitations of the study

- ESPNcricinfo Statsguru provides a vast collection of historical data, but gathering every data point was not that easy. Hence, just the batting, bowling, and team statistics tables from Statsguru were scraped, and appropriate features were chosen by considering previous works.
- Since tweets containing questions were not useful, a regex pattern was introduced to exclude those types of tweets from the study. Tweets containing question words such as why, what, and how were identified and removed using that pattern, although not all tweets containing those terms may actually pose questions. So, employing this pattern can increase the possibility of missing important tweets.
- When building the fans' prediction score variables, only a few written prediction patterns were considered to make the process simple.
- Due to the time and cost limitations of data labeling, only 3000 tweets were used to train and evaluate the sentiment analysis models' performances.
- Due to the unavailability and shortage of tweets, only a short time period (from 2011-01-01 to 2022-09-14) was considered for collecting data.

7.4 Suggestions for future work

It would be better to consider the following suggestions when creating the datasets:

- Adding some more features to the historical dataset
- Labeling more tweets to train the sentiment analysis model
- Adding new features to the Twitter dataset and considering not only the tweets but also the metrics such as retweet count and likes that can be used to measure how popular and engaged a tweet is

Deep learning techniques can be applied to create final prediction models to enhance performance. Creating a web or mobile application for the final prediction will be helpful for interested parties. A prediction model like this using both historical and Twitter data can be created for other cricket formats and sports as well.

7.5 Conclusion

To achieve the main objective of this study, three models were built. The combined model outperformed each separate model with a 73.7% F1 score, and the Twitter model performed better than the Historical model. Since the model used data that were collected one hour before the match, this model can be used to predict the match outcome even before the toss. This can be really helpful for team management when deciding their final squad. Checking whether the Twitter-derived features can provide useful information on top of the historical data to predict the match outcome is one of the secondary objectives of this study. Since the performance improved when the two datasets were combined, it can be stated that Twitter data have information that historical data do not have in predicting T20I match outcomes. Not only that, but the fans' sentiment score variables for the two teams were the most important variables in the best model, the Historical+Twitter model. Altogether, it can be concluded that Twitter-derived features provided useful information on top of the historical data for this study. To calculate values for those two sentiment score variables, building a model to identify the sentiment of pre-match tweets was needed. That was another secondary objective of this study. With a 95.3% F1 Score for the sentiment analysis, the transfer learning-RoBERTa-based model surpassed VADER and LSTM+GloVe. Beyond that, to identify the sentiment of a tweet, a web application was created using Streamlit. To focus on the final secondary objective, the best model was applied to the data from the T20 World Cup 2022. It was found that this model could outperform bookmakers' forecasts and generate profits.

Appendices

Appendix A : Snapshots of the datasets

#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Venue	T1_Mat	T1_AvgRunsScore	T1_AvgRunsConceded	T1_AvgWktsTaken	T1_AvgWktsLost	T1_AvgBound	T1_W/L	T2_Mat	T2_AvgRunsScore	T2_AvgRunsConceded	T2_AvgWktsTaken	T2_AvgWktsLost	T2_AvgBound	T2_W/L	Winner
2	Africa	34	7.1	7.27	0.33	0.32	0.85	1.833	22	7.69	7.99	0.28	0.32	0.95	1	0
3	Oceania	36	8.01	7.38	0.35	0.33	0.98	1.4	30	7.56	7.7	0.32	0.32	0.93	1.142	0
4	Oceania	37	8	7.39	0.35	0.33	0.98	1.312	31	7.56	7.7	0.32	0.33	0.93	1.214	1
5	America	38	7.14	7.18	0.33	0.34	0.84	1.235	24	7.13	8.08	0.29	0.36	0.89	0.714	0
6	America	23	7.71	7.94	0.28	0.32	0.96	1.09	25	7.13	8.04	0.29	0.36	0.89	0.785	1
7	Europe	32	7.53	7.68	0.32	0.33	0.91	1.133	29	7.05	7.46	0.3	0.33	0.86	1.071	0
8	Asia	38	7.98	7.38	0.35	0.33	0.97	1.375	30	7.05	7.43	0.31	0.32	0.86	1.142	0
9	Asia	39	7.98	7.44	0.34	0.33	0.97	1.294	31	7.13	7.45	0.31	0.32	0.88	1.214	0
10	Europe	24	7.69	7.9	0.28	0.32	0.95	1.181	33	7.49	7.68	0.31	0.33	0.9	1.062	0
11	Europe	34	7.51	7.69	0.32	0.33	0.91	1.125	26	7.12	8.04	0.29	0.36	0.89	0.733	1
12	Europe	35	7.51	7.65	0.32	0.32	0.91	1.187	27	7.08	8.05	0.28	0.36	0.88	0.687	0
13	Asia	28	7.02	7.95	0.29	0.36	0.87	0.75	13	6.22	8.22	0.25	0.46	0.73	0.083	0
14	Africa	35	7.1	7.3	0.33	0.32	0.85	1.692	40	7.95	7.45	0.35	0.33	0.97	1.222	0
15	Africa	36	7.09	7.3	0.33	0.32	0.85	1.571	41	7.93	7.44	0.35	0.33	0.96	1.277	1
16	Asia	36	7.45	7.59	0.32	0.33	0.9	1.117	25	7.69	7.92	0.28	0.32	0.95	1.083	1
17	Asia	32	7.13	7.44	0.31	0.32	0.88	1.285	39	7.12	7.19	0.33	0.35	0.83	1.166	0
18	Asia	40	7.12	7.19	0.33	0.34	0.83	1.222	14	6.24	8.07	0.26	0.45	0.72	0.166	1
19	Oceania	42	7.9	7.44	0.35	0.33	0.96	1.21	26	7.62	7.86	0.27	0.33	0.94	1	1
20	Oceania	43	7.91	7.43	0.35	0.33	0.96	1.263	27	7.58	7.88	0.27	0.33	0.93	0.928	0
21	Oceania	37	7.09	7.3	0.33	0.33	0.85	1.642	36	6.82	7.52	0.31	0.38	0.8	0.714	0
22	Oceania	38	7.09	7.3	0.32	0.32	0.85	1.533	37	6.83	7.51	0.31	0.38	0.79	0.761	1
23	Oceania	39	7.16	7.33	0.32	0.32	0.86	1.6	38	6.86	7.58	0.31	0.37	0.8	0.727	1
24	Asia	41	7.09	7.1	0.33	0.34	0.82	1.277	37	7.41	7.54	0.32	0.32	0.89	1.176	1
25	Asia	42	7.08	7.09	0.33	0.34	0.82	1.333	38	7.38	7.52	0.32	0.32	0.89	1.111	0
26	Asia	43	7.05	7.09	0.33	0.35	0.82	1.263	39	7.37	7.48	0.32	0.32	0.88	1.166	0

Appendix Figure A-1 – Snapshot of the finalized Historical Dataset

#	A	B
1	Date	Tweet
2	2011-10-13 13:14:31+00:00	big t20 game tonight, go the aussies!! #ausvsa #savaus
3	2011-10-13 10:40:05+00:00	just set the alarm for the cricket early hours of the morning. good luck to the @cricketaus go well lads. #thebulldoesntsleep
4	2011-10-13 10:16:04+00:00	@cricketaus : come on aussies!!show them whose de boss!!1b
5	2011-10-13 09:44:23+00:00	@cricketaus many congratulationssssssss @mitchmarsh235 :-))) .. good luck for the #sa series lads !
6	2011-10-13 09:28:35+00:00	good luck @cricketaus in the t20 tonight against sa! special shout out to @matthewwade13 on debut, go well! fair chance i'll be awake at 3am
7	2011-10-13 08:34:47+00:00	@cricketaus congrats @mitchmarsh235
8	2011-10-13 07:35:53+00:00	@cricketaus "i recruited robbie deans" - pat howard. wow aus cricket is saved
9	2011-10-13 06:36:39+00:00	all the very very best @davidwarner31. @stevesmith49 give a bang onnn performance boys. #cricketaus.
10	2011-10-13 06:13:27+00:00	well done pat howard! great choice by @cricketaus and a truly great man.
11	2011-10-13 04:05:54+00:00	like the choice of pat howard as gm of team performance by @cricketaus has the sporting background but not the ties/baggage of cricket past
12	2011-10-13 03:36:18+00:00	good decision @cricketaus to appoint former @qantaswallabies player and hp manager, pat howard to a newly created hp role.
13	2011-10-13 02:01:41+00:00	cameron white calls for t20 shake-up http://t.co/rb1oe67w @cricketaus
14	2011-10-12 16:26:09+00:00	i felt never excited about t20 matches,, but the due good wishes to australia t20 side for tomorrow #ausvsa

Appendix Figure A-2 – Snapshot of tweets scraped using Snsrape

	A	B	C	D	E	F	G
1	T1_TwitterVol	T2_TwitterVol	T1_FansSent	T2_FansSent	T1_FansPred	T2_FansPred	Winner
53	0.487	0.513	0.317	0.494	0.611	0.389	0
54	0.409	0.591	0.318	0.500	0.000	0.000	0
55	0.201	0.799	0.173	0.624	0.222	0.778	0
56	0.351	0.649	0.268	0.516	0.000	1.000	0
57	0.813	0.187	0.691	0.164	1.000	0.000	0
58	0.538	0.462	0.497	0.434	0.333	0.667	0
59	0.506	0.494	0.428	0.444	0.667	0.333	0
60	0.192	0.808	0.115	0.769	0.000	1.000	1
61	0.057	0.943	0.052	0.747	0.500	0.500	0
62	0.926	0.074	0.878	0.061	1.000	0.000	1

Appendix Figure A-3 – Snapshot of the finalized Twitter dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Venue	T1_Mat	T1_AvgRur	T1_AvgRur	T1_AvgWk	T1_AvgWk	T1_AvgBoi	T1_W/L	T2_Mat	T2_AvgRur	T2_AvgRur	T2_AvgWk	T2_AvgWk	T2_AvgBoi	T2_W/L	T1_Twitter	T2_Twitter	T1_FansSet	T2_FansSet	T1_FansPr	T2_FansPr	Winner
53	Asia	51	6.91	7.03	0.33	0.35	0.81	1.318	34	7.53	7.86	0.28	0.3	0.92	1	0.487	0.513	0.317	0.494	0.611	0.389	0
54	Asia	47	7.36	7.41	0.31	0.32	0.88	1.238	38	6.99	7.34	0.31	0.34	0.86	1.111	0.409	0.591	0.318	0.5	0	0	0
55	Asia	51	7.84	7.35	0.34	0.33	0.95	1.318	52	6.89	7.04	0.33	0.35	0.8	1.26	0.201	0.799	0.173	0.624	0.222	0.778	0
56	Asia	46	7.25	7.35	0.32	0.32	0.87	1.555	35	7.52	7.82	0.29	0.3	0.92	1.058	0.351	0.649	0.268	0.516	0	1	0
57	Asia	53	6.9	7.01	0.33	0.35	0.8	1.304	39	7.01	7.34	0.31	0.34	0.86	1.166	0.813	0.187	0.691	0.164	1	0	0
58	Asia	52	7.8	7.35	0.34	0.33	0.94	1.26	37	7.23	7.93	0.28	0.34	0.91	0.761	0.538	0.462	0.497	0.434	0.333	0.667	0
59	Asia	40	7	7.31	0.31	0.33	0.86	1.222	38	7.29	7.92	0.29	0.34	0.93	0.809	0.506	0.494	0.428	0.444	0.667	0.333	0
60	Asia	39	7.27	7.85	0.29	0.34	0.92	0.857	17	6.2	8.14	0.25	0.44	0.73	0.133	0.192	0.808	0.115	0.769	0	1	1
61	Asia	48	7.36	7.43	0.31	0.32	0.88	1.181	36	7.51	7.81	0.3	0.3	0.92	1.117	0.057	0.943	0.052	0.747	0.5	0.5	0
62	Africa	47	7.25	7.35	0.31	0.32	0.87	1.473	45	6.96	7.7	0.3	0.37	0.82	0.666	0.926	0.074	0.878	0.061	1	0	1
63	Asia	37	7.52	7.81	0.3	0.3	0.92	1.176	49	7.36	7.45	0.31	0.32	0.88	1.13	0.98	0.02	0.904	0.015	1	0	0
64	Africa	48	7.25	7.29	0.32	0.32	0.88	1.526	46	6.91	7.69	0.3	0.37	0.82	0.642	0.802	0.198	0.704	0.173	0	0	0
65	Asia	54	6.87	7.01	0.33	0.35	0.8	1.25	38	7.53	7.83	0.29	0.3	0.92	1.111	0.385	0.615	0.316	0.531	0.421	0.579	1
66	Africa	49	7.28	7.31	0.31	0.32	0.88	1.45	47	6.93	7.71	0.29	0.37	0.83	0.678	0.86	0.14	0.72	0.108	0	0	1
67	Asia	55	6.87	7	0.33	0.35	0.8	1.291	39	7.5	7.8	0.29	0.3	0.91	1.052	0.412	0.588	0.31	0.516	0.4	0.6	0
68	Oceania	53	7.79	7.4	0.34	0.33	0.94	1.208	41	6.95	7.29	0.31	0.34	0.85	1.157	0.826	0.174	0.733	0.151	1	0	0
69	Oceania	42	6.96	7.28	0.31	0.34	0.85	1.21	54	7.77	7.4	0.34	0.33	0.93	1.16	0.237	0.763	0.186	0.492	0.5	0.5	1
70	Oceania	50	7.39	7.48	0.31	0.32	0.88	1.173	48	6.94	7.74	0.3	0.37	0.83	0.655	0.358	0.642	0.328	0.627	0	0	1
71	Oceania	51	7.45	7.49	0.31	0.32	0.9	1.217	49	6.95	7.8	0.3	0.37	0.83	0.633	0.571	0.429	0.5	0.357	1	0	0
72	Oceania	55	7.77	7.4	0.33	0.33	0.93	1.115	40	7.33	7.88	0.28	0.33	0.93	0.904	0.488	0.512	0.349	0.465	0	0	0
73	Oceania	52	7.43	7.53	0.31	0.32	0.9	1.166	50	7	7.78	0.3	0.37	0.84	0.666	0.706	0.294	0.706	0.294	1	0	1
74	Africa	50	7.3	7.31	0.32	0.32	0.88	1.5	56	6.9	7.04	0.32	0.35	0.81	1.24	0.352	0.648	0.341	0.602	0.5	0.5	0

Appendix Figure A-4– Snapshot of the finalized Historical+Twitter dataset

Appendix B : Sample of Python codes

AUSvSL 2013-01-26

```
# Team1 = AUS
T1_bat = pd.read_html("https://stats.espncricinfo.com/ci/engine/stats/index.html?class=3;filter=advanced;groupby=team;opposition=
T1_bat = T1_bat.add_prefix('T1_')
T1_bat = T1_bat.add_suffix('_bat')

T1_bowl = pd.read_html("https://stats.espncricinfo.com/ci/engine/stats/index.html?class=3;filter=advanced;groupby=team;opposition=
T1_bowl = T1_bowl.add_prefix('T1_')
T1_bowl = T1_bowl.add_suffix('_bowl')

T1_team = pd.read_html("https://stats.espncricinfo.com/ci/engine/stats/index.html?class=3;filter=advanced;opposition=1;opposition=
T1_team = T1_team.add_prefix('T1_')
T1_team = T1_team.add_suffix('_team')

T1 = pd.concat([T1_bat, T1_bowl, T1_team], axis=1)

# Team2 = SL
T2_bat = pd.read_html("https://stats.espncricinfo.com/ci/engine/stats/index.html?class=3;filter=advanced;groupby=team;opposition=
T2_bat = T2_bat.add_prefix('T2_')
T2_bat = T2_bat.add_suffix('_bat')

T2_bowl = pd.read_html("https://stats.espncricinfo.com/ci/engine/stats/index.html?class=3;filter=advanced;groupby=team;opposition=
T2_bowl = T2_bowl.add_prefix('T2_')
T2_bowl = T2_bowl.add_suffix('_bowl')

T2_team = pd.read_html("https://stats.espncricinfo.com/ci/engine/stats/index.html?class=3;filter=advanced;opposition=1;opposition=
T2_team = T2_team.add_prefix('T2_')
T2_team = T2_team.add_suffix('_team')

T2 = pd.concat([T2_bat, T2_bowl, T2_team], axis=1)

M = pd.concat([T1, T2], axis=1)

# save as a csv file
M.to_csv('historical.csv', mode='a', index=False, header=True)
```

Appendix Figure B-1– Historical data scraping code sample

```
# ENG
# call the dataset
data = pd.read_csv('tweets 1.0/2011-01-12 AUSvsENG/ENG-2011-01-12.csv')
# convert Tweets to Lowercase
data['Tweet'] = data['Tweet'].str.lower()
# remove duplicates
df = data.drop_duplicates(subset=['Tweet'], keep='first')
# call the function has_question_word
sentences = df['Tweet'].to_list()
Q=[]
for sentence in sentences:
    if has_question_word(sentence):
        Q.append(sentence)

df = df.drop(df[df['Tweet'].isin(Q)].index)
# call the function select_tweet_with_item
tweets = df['Tweet']
list1 = ENGvAUS
I=[]
for tweet in tweets:
    if not select_tweet_with_item(tweets, list1):
        I.append(tweet)

df = df.drop(df[df['Tweet'].isin(I)].index)

df.head()
# save the dataset
df.to_csv('tweets 2.0/2011-01-12 AUSvsENG/ENG-2011-01-12.csv')
```

```

# ENG
# call the dataset
data = pd.read_csv('tweets 1.0/2011-01-12 AUSvsENG/ENG-2011-01-12.csv')
# convert Tweets to Lowercase
data['Tweet'] = data['Tweet'].str.lower()
# remove duplicates
df = data.drop_duplicates(subset=['Tweet'], keep='first')
# call the function has_question_word
sentences = df['Tweet'].to_list()
Q=[]
for sentence in sentences:
    if has_question_word(sentence):
        Q.append(sentence)

df = df.drop(df[df['Tweet'].isin(Q)].index)
# call the function select_tweet_with_item
tweets = df['Tweet']
list1 = ENGVAUS
I=[]
for tweet in tweets:
    if not select_tweet_with_item(tweets, list1):
        I.append(tweet)

df = df.drop(df[df['Tweet'].isin(I)].index)

df.head()
# save the dataset
df.to_csv('tweets 2.0/2011-01-12 AUSvsENG/ENG-2011-01-12.csv')

```

Appendix Figure B-2- Twitter data scraping code sample

```

st.header('Cric-Tweets')
sentiment_model = pipeline(model="sppm/cric-tweets-sentiment-analysis")
with st.expander('Analyze Tweet'):
    text = st.text_input('Tweet here: ')
    if text:
        senti = list(sentiment_model(text)[0].values())[0]
        if senti=='LABEL_1':
            label = senti.replace(senti, 'Positive')
        elif senti=='LABEL_0':
            label = senti.replace(senti, 'Negative')
        st.write('Label: ', label)
with st.expander('Analyze CSV'):
    upl = st.file_uploader('Upload file: ')
    if upl:
        df = pd.read_csv(upl, encoding='latin1')
        del df['Unnamed: 0']
        b=[]
        for i in range(len(df)):
            b.append(list(sentiment_model(df['Tweet'][i])[0].values())[0])
        df['Sentiment'] = b
        # Convert label1/label2 string columns into int columns of 1/0
        df[['Sentiment']] = \
            (df[['Sentiment']] == 'LABEL_1').astype(int)
        st.write(df.head())
        @st.cache
        def convert_df(df):
            return df.to_csv().encode('utf-8')
        csv = convert_df(df)
        st.download_button(
            label="Download",
            data=csv,
            file_name= upl.name,
            mime='text/csv',
        )

team = upl.name.split('--')[0]
if team == 'AUS':
    words = ['Australia will win', 'Australia win', 'Aus win', 'Aus will win']
elif team == 'BAN':
    words = ['Bangladesh will win', 'Bangladesh win', 'Ban win', 'Ban will win']
elif team == 'ENG':
    words = ['England will win', 'England win', 'Eng win', 'Eng will win']
elif team == 'IND':
    words = ['India will win', 'India win', 'Ind win', 'Ind will win']
elif team == 'NZ':
    words = ['New Zealand will win', 'New Zealand win', 'NZ win', 'NZ will win']
elif team == 'PAK':
    words = ['Pakistan will win', 'Pakistan win', 'Pak win', 'Pak will win']
elif team == 'SA':
    words = ['South Africa will win', 'South Africa win', 'SA win', 'SA will win']
elif team == 'SL':
    words = ['Sri Lanka will win', 'Sri Lanka win', 'SL win', 'SL will win']
elif team == 'WI':
    words = ['West Indies will win', 'West Indies win', 'WI win', 'WI will win']

df['Tweet'] = df['Tweet'].str.lower()
sentence = df['Tweet'].tolist()
words = [word.lower() for word in words]

res = [any([k in s for k in words]) for s in sentence]
output = [sentence[i] for i in range(0, len(res)) if res[i]]

st.write('Winner prediction count: ', len(output))
st.write('Number of Positive Tweets: ', df.Sentiment.sum())
st.write('Total Number of Tweets: ', len(df))

```

Appendix Figure B-3-Code for the web application

```

df = pd.read_csv("historical-correct.csv")
# remove unwanted columns
df1 = df.drop(['Date', 'T1', 'T2'], axis=1)
# Convert T1/T2 string columns into int columns of 1/0
df1[['Winner']] = \
(df1[['Winner']] == 'T1').astype(int)
# create overs batted var
df1['T1_OversBat'] = 0
for i in range(0, len(df1)):
    if str(round((df1['T1_BowlsFace'][i]/6)-int(df1['T1_BowlsFace'][i]/6),3))[1:] == '.167' :
        df1['T1_OversBat'][i] = (df1['T1_BowlsFace'][i]/6)
    elif str(round((df1['T1_BowlsFace'][i]/6)-int(df1['T1_BowlsFace'][i]/6),3))[1:] == '.333' :
        df1['T1_OversBat'][i] = (df1['T1_BowlsFace'][i]/6)
    elif str(round((df1['T1_BowlsFace'][i]/6)-int(df1['T1_BowlsFace'][i]/6),3))[1:] == '.5' :
        df1['T1_OversBat'][i] = (df1['T1_BowlsFace'][i]/6)
    elif str(round((df1['T1_BowlsFace'][i]/6)-int(df1['T1_BowlsFace'][i]/6),3))[1:] == '.667' :
        df1['T1_OversBat'][i] = (df1['T1_BowlsFace'][i]/6)
    elif str(round((df1['T1_BowlsFace'][i]/6)-int(df1['T1_BowlsFace'][i]/6),3))[1:] == '.833' :
        df1['T1_OversBat'][i] = (df1['T1_BowlsFace'][i]/6)
    elif str(round((df1['T1_BowlsFace'][i]/6)-int(df1['T1_BowlsFace'][i]/6),3))[1:] == '.0' :
        df1['T1_OversBat'][i] = (df1['T1_BowlsFace'][i]/6)
df1['T2_OversBat'] = 0
for i in range(0, len(df1)):
    if str(round((df1['T2_BowlsFace'][i]/6)-int(df1['T2_BowlsFace'][i]/6),3))[1:] == '.167' :
        df1['T2_OversBat'][i] = (df1['T2_BowlsFace'][i]/6)
    elif str(round((df1['T2_BowlsFace'][i]/6)-int(df1['T2_BowlsFace'][i]/6),3))[1:] == '.333' :
        df1['T2_OversBat'][i] = (df1['T2_BowlsFace'][i]/6)
    elif str(round((df1['T2_BowlsFace'][i]/6)-int(df1['T2_BowlsFace'][i]/6),3))[1:] == '.5' :
        df1['T2_OversBat'][i] = (df1['T2_BowlsFace'][i]/6)
    elif str(round((df1['T2_BowlsFace'][i]/6)-int(df1['T2_BowlsFace'][i]/6),3))[1:] == '.667' :
        df1['T2_OversBat'][i] = (df1['T2_BowlsFace'][i]/6)
    elif str(round((df1['T2_BowlsFace'][i]/6)-int(df1['T2_BowlsFace'][i]/6),3))[1:] == '.833' :
        df1['T2_OversBat'][i] = (df1['T2_BowlsFace'][i]/6)
    elif str(round((df1['T2_BowlsFace'][i]/6)-int(df1['T2_BowlsFace'][i]/6),3))[1:] == '.0' :
        df1['T2_OversBat'][i] = (df1['T2_BowlsFace'][i]/6)

```

```

for i in range(0, len(df1)):
    if df1['Venue'][i] in Oceania:
        df1['Venue'][i] = 'Oceania'
    elif df1['Venue'][i] in Africa:
        df1['Venue'][i] = 'Africa'
    elif df1['Venue'][i] in Asia:
        df1['Venue'][i] = 'Asia'
    elif df1['Venue'][i] in Europe:
        df1['Venue'][i] = 'Europe'
    elif df1['Venue'][i] in America:
        df1['Venue'][i] = 'America'
# 2. avg runs scored per over var
df1['T1_AvgRunsScored/Over'] = df1.apply(lambda row: np.round(row.T1_RunsScored/row.T1_OversBat,2) , axis = 1)
df1['T2_AvgRunsScored/Over'] = df1.apply(lambda row: np.round(row.T2_RunsScored/row.T2_OversBat,2) , axis = 1)
# 3. avg runs conceded per over var
df1['T1_AvgRunsConceded/Over'] = df1.apply(lambda row: np.round(row.T1_RunsConceded/row.T1_OversBowled,2) , axis = 1)
df1['T2_AvgRunsConceded/Over'] = df1.apply(lambda row: np.round(row.T2_RunsConceded/row.T2_OversBowled,2) , axis = 1)
# 4. create avg boundaries earned per over var
df1['T1_Bound'] = df1.apply(lambda row: np.round((row.T1_4s+row.T1_6s)/row.T1_OversBat,2) , axis = 1)
df1['T2_Bound'] = df1.apply(lambda row: np.round((row.T2_4s+row.T2_6s)/row.T2_OversBat,2) , axis = 1)
# 5. avg wickets taken per over
df1['T1_Avg_Wkts_Taken'] = df1.apply(lambda row: np.round(row.T1_WktsTaken/row.T1_OversBowled,2) , axis = 1)
df1['T2_Avg_Wkts_Taken'] = df1.apply(lambda row: np.round(row.T2_WktsTaken/row.T2_OversBowled,2) , axis = 1)
# 6. avg wickets lost
df1['T1_Avg_Wkts_Lost'] = df1.apply(lambda row: np.round(row.T1_WktsLost/row.T1_OversBat,2) , axis = 1)
df1['T2_Avg_Wkts_Lost'] = df1.apply(lambda row: np.round(row.T2_WktsLost/row.T2_OversBat,2) , axis = 1)
# remove unwanted columns
data = df1.drop([
    'T1_BowlsFace', 'T1_OversBowled',
    'T1_RunsScored', 'T1_4s', 'T1_6s', 'T1_RunsConceded', 'T1_WktsTaken',
    'T1_WktsLost', 'T2_BowlsFace', 'T2_OversBowled',
    'T2_RunsScored', 'T2_4s', 'T2_6s', 'T2_RunsConceded', 'T2_WktsTaken',
    'T2_WktsLost', 'T1_OversBat', 'T2_OversBat',
    'T1_OversBowled', 'T2_OversBowled',
], axis=1)
# save the dataset
data.to_csv('historical-featured.csv')

```

Appendix Figure B-4- Python code for Historical dataset creation


```

df = pd.read_csv("tweets-dataset.csv", encoding='latin1')
# Convert T1/T2 string columns into int columns of 1/0
df[['winner']] = \
(df[['winner']] == 'T1').astype(int)
# create Twitter Volume var
df['T1_Twitter_Vol'] = df.apply(lambda row: (row.T1_Tweets/(row.T1_Tweets+row.T2_Tweets)), axis = 1)
df['T2_Twitter_Vol'] = df.apply(lambda row: (row.T2_Tweets/(row.T1_Tweets+row.T2_Tweets)), axis = 1)
# create Fans Sentiment var
df['T1_Fans_Sent'] = df.apply(lambda row: (row.T1_Positive_Tweets/(row.T1_Tweets+row.T2_Tweets)), axis = 1)
df['T2_Fans_Sent'] = df.apply(lambda row: (row.T2_Positive_Tweets/(row.T1_Tweets+row.T2_Tweets)), axis = 1)
# create Fans pred var
df['T1_Fans_Pred'] = df.apply(lambda row: (row.T1_Winner_Prediction/(row.T1_Winner_Prediction+row.T2_Winner_Prediction)) if (r
df['T2_Fans_Pred'] = df.apply(lambda row: (row.T2_Winner_Prediction/(row.T1_Winner_Prediction+row.T2_Winner_Prediction)) if (r
# remove unwanted columns
data = df.drop(['T1_Tweets', 'T1_Positive_Tweets', 'T1_Winner_Prediction',
               'T2_Tweets', 'T2_Positive_Tweets', 'T2_Winner_Prediction'], axis=1)
# save the dataset
data.to_csv('twitter-featured.csv')

```

Appendix Figure B-5- Python code for Twitter dataset creation

```

# 1. target variable distribution
fig, ax1 = plt.subplots(1,1)
g = sns.countplot(x=train['winner'],ax=ax1)
for p in g.patches:
    height = p.get_height()
    g.text(x = p.get_x() + (p.get_width()/2),
           y = height + 0.3,
           s = height,
           ha = 'center',
           fontsize=8,
           )
plt.xlabel('winner', fontsize = 10)
plt.ylabel('Count', fontsize = 10)
g.set_xticklabels(['Team2', 'Team1'], fontsize = 8)
plt.show()

# 2. dython plot
nums = [
    'T1_Mat', 'T1_AvgRunsScored',
    'T1_AvgRunsConceded', 'T1_AvgWktsTaken', 'T1_AvgWktsLost',
    'T1_AvgBound', 'T1_W/L', 'T2_Mat', 'T2_AvgRunsScored',
    'T2_AvgRunsConceded', 'T2_AvgWktsTaken', 'T2_AvgWktsLost',
    'T2_AvgBound', 'T2_W/L'
]
cats = ['Venue']
from dython.nominal import associations
associations(train.drop(['winner'], axis=1),
             nominal_columns = cats,
             numerical_columns = nums,
             fmt='.2f', cmap='Blues_r',
             mark_columns = True,
             figsize = (10,10))

# 3. PCA plot
train1 = train.drop(['Venue'], axis=1)
pca = PCA()
pipe = Pipeline([('scaler', StandardScaler()), ('pca', pca)])
X = pipe.fit_transform(train1.drop(['winner'], axis=1))
components = pca.fit_transform(X)

total_var = pca.explained_variance_ratio_.sum() * 100
fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=train1['winner'],
    title=f'Total Explained Variance: {total_var:.2f}%',
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'})
fig.show()

# 4. Factor Analysis
train_fa = train.drop(['Venue', 'winner'], axis=1)
## 4.1 bartlett test
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value,p_value=calculate_bartlett_sphericity(train_fa)
chi_square_value, p_value
## 4.2 kmo test
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all,kmo_model=calculate_kmo(train_fa)
kmo_model
## 4.3 scree plot
from factor_analyzer import FactorAnalyzer
fa = FactorAnalyzer(rotation = None,impute = "drop",
                    n_factors=train_fa.shape[1])
fa.fit(train_fa)
ev,_ = fa.get_eigenvalues()
plt.scatter(range(1,train_fa.shape[1]+1),ev)
plt.plot(range(1,train_fa.shape[1]+1),ev)
plt.title('Scree Plot')
plt.xlabel('Factors')
plt.ylabel('Eigen Value')
plt.grid()
## 4.4 factor loadings
fa = FactorAnalyzer(n_factors=5,rotation='varimax')
fa.fit(train_fa)
df = pd.DataFrame(fa.loadings_,index=train_fa.columns)
df.rename({0: 'Factor 1', 1: 'Factor 2', 2:'Factor 3', 3: 'Factor 4', 4:'Factors
         }, axis=1, inplace=True)
print(df)

```

Appendix Figure B-6- Sample code for data analysis

```

# Load data
df = pd.read_csv('test.csv', encoding = 'latin1')
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm
sia = SentimentIntensityAnalyzer()
# Create Prediction column based on Polarity Score
df['Prediction'] = df['Tweet'].apply(lambda x: 1 if sia.polarity_scores(x)['compound'] >= 0 else -1)
# TP, TN, FP, FN
def conf_matrix(x):
    if x[6] == 1 and x[7] == 1:
        return 'TP'
    elif x[6] == 1 and x[7] == 0:
        return 'FN'
    elif x[6] == 0 and x[7] == 1:
        return 'FP'
    elif x[6] == 0 and x[7] == 0:
        return 'TN'
    else:
        return 0
df['Conf_Matrix'] = df.apply(lambda x: conf_matrix(x), axis=1)
# calculate metrics
conf_vals = df.Conf_Matrix.value_counts().to_dict()
print(conf_vals)
accuracy = (conf_vals['TP'] + conf_vals['TN']) / (conf_vals['TP'] + conf_vals['TN'] + conf_vals['FP'] + conf_vals['FN'])
precision = conf_vals['TP'] / (conf_vals['TP'] + conf_vals['FP'])
recall = conf_vals['TP'] / (conf_vals['TP'] + conf_vals['FN'])
f1_score = 2*precision*recall / (precision + recall)
print('Accuracy: ', round(100 * accuracy, 2), '%',
      '\nPrecision: ', round(100 * precision, 2), '%',
      '\nRecall: ', round(100 * recall, 2), '%',
      '\nF1 Score: ', round(100 * f1_score, 2), '%')

```

Appendix Figure B-7-Code for VADER sentiment analysis

```

# load the dataset
df = pd.read_csv('tweets-sample.csv', encoding='latin1')
# data cleaning
df['CleanTweet'] = df['Tweet'].apply(lambda x: nfx.remove_hashtags(x))
df['CleanTweet'] = df['CleanTweet'].apply(lambda x: nfx.remove_userhandles(x))
df['CleanTweet'] = df['CleanTweet'].apply(nfx.remove_multiple_spaces)
df['CleanTweet'] = df['CleanTweet'].apply(nfx.remove_urls)
df['CleanTweet'] = df['CleanTweet'].apply(nfx.remove_puncts)
# download and unzip glove embedding from stanford
!wget http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
!unzip glove.6B.zip
words = dict()
def add_to_dict(d, filename):
    with open(filename, 'r') as f:
        for line in f.readlines():
            line = line.split(' ')

            try:
                d[line[0]] = np.array(line[1:], dtype=float)
            except:
                continue
add_to_dict(words, 'glove.6B.50d.txt')
# tokenizing
tokenizer = nltk.RegexpTokenizer(r"\w+")
# lemmatization
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
def message_to_useful_tokens(s):
    tokens = tokenizer.tokenize(s)
    lemmatized_tokens = [lemmatizer.lemmatize(t) for t in tokens]
    useful_tokens = [t for t in lemmatized_tokens if t in words]
    return useful_tokens

plt.hist(sequence_lengths) # majority in between 0 & 20
plt.xlabel('Number of tokens', fontsize = 10)
plt.ylabel('Count', fontsize = 10)
plt.show()
pd.Series(sequence_lengths).describe()
# padding
from copy import deepcopy
def pad_X(X, desired_sequence_length=50):
    X_copy = deepcopy(X)
    for i, x in enumerate(X):
        x_seq_len = x.shape[0]
        sequence_length_difference = desired_sequence_length - x_seq_len
        pad = np.zeros(shape=(sequence_length_difference, 50))
        X_copy[i] = np.concatenate([x, pad])
    return np.array(X_copy).astype(float)
X_train = pad_X(X_train)
X_val, y_val = df.to_X_y(val_df)
X_val = pad_X(X_val)
X_test, y_test = df.to_X_y(test_df)
X_test = pad_X(X_test)
# LSTM architecture
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
model = Sequential([
    model.add(layers.Input(shape=(50, 50)))
    model.add(layers.LSTM(128, return_sequences=True))
    model.add(layers.Dropout(0.2))
    model.add(layers.LSTM(64, return_sequences=True))
    model.add(layers.Dropout(0.2))
    model.add(layers.LSTM(64, return_sequences=True))
    model.add(layers.Dropout(0.2))
    model.add(layers.Flatten())
    model.add(layers.Dense(1, activation='sigmoid'))

```

```

# word embedding to convert a word to set of numbers
def message_to_word_vectors(message, word_dict=words):
    processed_list_of_tokens = message_to_useful_tokens(message)
    vectors = []
    for token in processed_list_of_tokens:
        if token not in word_dict:
            continue
        token_vector = word_dict[token]
        vectors.append(token_vector)
    return np.array(vectors, dtype=float)

# train, validation, test split
df = df.sample(frac=1, random_state=1)
df.reset_index(drop=True, inplace=True)
split_index_1 = int(len(df) * 0.7)
split_index_2 = int(len(df) * 0.85)
train_df, val_df, test_df =
df[:split_index_1], df[split_index_1:split_index_2], df[split_index_2:]

# split X & y
def df_to_X_y(df):
    y = df['label'].to_numpy().astype(int)
    all_word_vector_sequences = []
    for message in df['clean_tweet']:
        message_as_vector_seq = message_to_word_vectors(message)
        if message_as_vector_seq.shape[0] == 0:
            message_as_vector_seq = np.zeros(shape=(1, 50))
        all_word_vector_sequences.append(message_as_vector_seq)
    return all_word_vector_sequences, y

# split X_train, y_train
X_train, y_train = df_to_X_y(train_df)
# identify max. sequence length
sequence_lengths = []
for i in range(len(X_train)):
    sequence_lengths.append(len(X_train[i]))
import matplotlib.pyplot as plt

# function for f1-score
def f1_score(y_true, y_pred):
    def recall(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
        recall = true_positives / (possible_positives + K.epsilon())
        return recall
    def precision(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + K.epsilon())
        return precision
    precision = precision(y_true, y_pred)
    recall = recall(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))

# model fit
from tensorflow.keras.losses import BinaryCrossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
cp = ModelCheckpoint('model/', save_best_only=True)
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss=BinaryCrossentropy(),
              metrics=[binary_accuracy, f1_score])

weights =
{0: frequencies.sum() / frequencies[0], 1: frequencies.sum() / frequencies[1]}
from keras import callbacks
earlystopping = callbacks.EarlyStopping(monitor="val_loss",
                                       mode="min", patience = 5,
                                       restore_best_weights = True)
history = model.fit(X_train, y_train, validation_data=(X_val, y_val),
                   epochs=200,
                   class_weight=weights, callbacks=[earlystopping, cp])

```

Appendix Figure B-8- Code for LSTM sentiment analysis

```

MODEL = "cardiffnlp/twitter-roberta-base-sentiment-latest"
MAX_TRAINING_EXAMPLES = -1
seed = 223
set_seed(seed)
# load datasets
data_files = {"train": "train.csv", "val": "val.csv", "test": "test.csv"}
dataset = load_dataset("csv", data_files=data_files, encoding='latin1')
dataset

# use model's tokenizer to get text encodings
tokenizer = AutoTokenizer.from_pretrained(MODEL, use_fast=True)
dataset = dataset.map(lambda e: tokenizer(e['tweet'], truncation=True),
                    batched=True)

# make sure to use whole train dataset if MAX_TRAINING_EXAMPLES == -1
if MAX_TRAINING_EXAMPLES == -1: MAX_TRAINING_EXAMPLES = dataset['train'].shape[0]
# split into train/val/test sets
train_dataset = dataset['train'].select(range(MAX_TRAINING_EXAMPLES))
val_dataset = dataset['val']
test_dataset = dataset['test']

# take number of labels
num_labels =
len(set(train_dataset['labels'])) if 'labels' in train_dataset.features.keys()
else len(set(train_dataset['label']))
model =
AutoModelForSequenceClassification.from_pretrained(MODEL,
                                                  ignore_mismatched_sizes=True,
                                                  num_labels=num_labels)

# Log in to Hugging Face account and get API token
from huggingface_hub import notebook_login
notebook_login()
repo_name = "cric-tweets-sentiment-analysis"

# define training arguments
training_args = TrainingArguments(
    output_dir=repo_name,
    learning_rate=5e-5,
    num_train_epochs=200,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    warmup_steps=100,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=160,
    evaluation_strategy='steps',
    eval_steps=160,
    load_best_model_at_end=True,
    save_steps=160,
    seed=seed,
    push_to_hub=True
)

# train the model
trainer = Trainer(
    model=model,
    tokenizer=tokenizer,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    callbacks=[EarlyStoppingCallback(3)],
)
trainer.train()

```

Appendix Figure B-9- Code for RoBERTa-based sentiment analysis

```

# define xtrain, ytrain, xtest, ytest
xtrain = train.drop(['winner'],axis=1)
ytrain = train['winner']
xtest = test.drop(['winner'],axis=1)
ytest = test['winner']
# convert categorical features using OneHotEncoding
encoded_xtrain = pd.get_dummies(data= xtrain,
                                columns=['Venue'])
encoded_xtest = pd.get_dummies(data= xtest,
                                columns=['Venue' ])
# standardized the dataset
st = StandardScaler()
xtrain = st.fit_transform( encoded_xtrain)
xtest = st.fit_transform( encoded_xtest)
# apply SMOTE
sm = SMOTE(random_state=42)
xtrain_sm, ytrain_sm = sm.fit_resample(xtrain,
                                       ytrain)

# 1. Logistic regression
model = LogisticRegression(random_state=42)
param_grid = [
    {'solver' : [ 'newton-cg', 'lbfgs', 'liblinear',
                  'sag', 'saga', 'newton-cholesky'],
     'penalty' : [ 'l1' ],
     # for Lasso - L1, ridge - L2, elasticnet - elasticnet
     'C' : [10 ],#1
     'max_iter' : [2000]
    }
]
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
                             random_state=42)
log_grid = GridSearchCV(model, param_grid= param_grid ,
                        verbose=True, n_jobs=-1, cv=cv , scoring='f1')
log_clf = log_grid.fit(xtrain_sm,ytrain_sm)

# 5. RF
model = RandomForestClassifier(random_state=42)
param_grid = [
    {'n_estimators' : [2000],
     'criterion': [ 'gini', 'entropy'],
     'max_features':['sqrt', 'log2', 0.2, 0.4],
     'min_samples_leaf': [ 70, 90, 120 ],
     'oob_score':[True],
     'min_samples_split':[5, 6, 7, 9]
    }
]
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
                             random_state=1)
rf_grid = GridSearchCV(estimator=model, param_grid= param_grid,
                       n_jobs=-1, cv=cv, scoring='f1',error_score='raise')
rf_clf = rf_grid.fit(xtrain_sm, ytrain_sm)

# 6. XGBoost
model = xgb.XGBClassifier(seed= 42)
param_grid = [
    {'learning_rate' :[1e-4],
     'n_estimators':[2000],
     'max_depth':[ 2, 4, 8],
     'min_child_weight':[5, 7, 9],
     'gamma':[0],
     'subsample':[0.3, 0.5, 0.7 ],
     'colsample_bytree':[ 0.85, 0.9 ],
     'objective': ['binary:logistic'],
     'nthread':[4],
     'scale_pos_weight':[1],
     'reg_alpha':[1e-5 ]
    }
]
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
                             random_state=1)
xgb_grid = GridSearchCV(estimator=model, param_grid= param_grid,
                        cv=cv, n_jobs= -1, scoring='f1',error_score='raise', verbose=0)
xgb_clf = xgb_grid.fit(xtrain_sm, ytrain_sm)

# 2. SVM
model = SVC(random_state=42)
param_grid = [
    {'kernel' : ['poly', 'rbf', 'sigmoid', 'linear' ],
     'gamma' :['scale', 'auto' ],
     'C': [0.1, 1, 10, 100]
    }
]
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
                             random_state=1)
svm_grid = GridSearchCV(model, param_grid=param_grid,
                        verbose=True, n_jobs=-1, cv=cv , scoring = 'f1', error_score='raise')
svm_clf = svm_grid.fit(xtrain_sm, ytrain_sm)

# 3. KNN
pipe = Pipeline([
    ('classifier', KNeighborsClassifier())
])
parameters = {
    'classifier__n_neighbors': [21],
    'classifier__p': [1, 2],
    'classifier__leaf_size': [1, 5, 10, 15],
    'classifier__weights': ['uniform' ],
    'classifier__metric' : [ 'manhattan', 'minkowski',
                           'euclidean']
}
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
                             random_state=1)
knn_grid = GridSearchCV(pipe, parameters,verbose=True, n_jobs=-1, cv=cv , scoring = 'f1', error_score='raise')
knn_clf = knn_grid.fit(xtrain_sm, ytrain_sm)

# 4. GNB
def set_random_seed(seed=100):
    np.random.seed(seed)
    random.seed(seed)
set_random_seed(100)
gnb_grid = GaussianNB(var_smoothing=10e-1 )
gnb_clf = gnb_grid.fit(xtrain_sm, ytrain_sm)

# 7. Voting classifier
voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rf_clf), ('svc', svm_clf),
                ('xgb', xgb_clf)],
    voting='hard')
voting_clf.fit(xtrain_sm, ytrain_sm )
# model evaluation
## on train set
y_predicted = log_clf.predict(xtrain_sm)
print('\nclassification report - logistic (l1) - train')
print(classification_report(ytrain_sm, y_predicted))
print('LogReg: ROC AUC = ',
      str(round(roc_auc_score(ytrain_sm, y_predicted)*100,1)), '%')
print('LogReg: Precision = ',
      str(round(precision_score(ytrain_sm, y_predicted)*100,1)), '%')
print('LogReg: Recall = ',
      str(round(recall_score(ytrain_sm, y_predicted)*100,1)), '%')
print('LogReg: Accuracy = ',
      str(round(accuracy_score(ytrain_sm, y_predicted)*100,1)), '%')
print('LogReg: F1-Score = ',
      str(round(f1_score(ytrain_sm, y_predicted)*100,1)), '%')
confusion_matrix(ytrain_sm, y_predicted)
## on test set
y_predicted = log_clf.predict( xtest)
print('\nclassification report - logistic(l1)+SMOTE - test ')
print(classification_report(ytest, y_predicted))
print('LogReg: ROC AUC = ',
      str(round(roc_auc_score(ytest, y_predicted)*100,1)), '%')
print('LogReg: Precision = ',
      str(round(precision_score(ytest, y_predicted)*100,1)), '%')
print('LogReg: Recall = ',
      str(round(recall_score(ytest, y_predicted)*100,1)), '%')
print('LogReg: Accuracy = ',
      str(round(accuracy_score(ytest, y_predicted)*100,1)), '%')
print('LogReg: F1-Score = ',
      str(round(f1_score(ytest, y_predicted)*100,1)), '%')
confusion_matrix(ytest, y_predicted)

```

Appendix Figure B-10- Codes for winner prediction model

Appendix C : Results of the factor score method

Appendix Table C-1- Results of the Historical model after applying factor score method

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Logistic Regression	52.8%	55.6%	54.5%	52.9%	55.0%
SVM	58.4%	60.6%	48.1%	58.4%	53.7%
KNN	59.9%	62.5%	49.4%	59.9%	55.2%
Naïve Bayes	57.0%	60.4%	52.7%	56.7%	56.3%
Random Forest	60.1%	59.9%	60.9%	60.1%	60.4%
XGBoost	64.8%	67.0%	58.4%	64.8%	62.4%

Appendix Table C-2- Results of the Historical+Twitter model after applying factor score method

Classifier	ROC AUC	Precision	Recall	Accuracy	F1 Score
Logistic Regression	61.1%	60.5%	63.8%	61.1%	62.1%
SVM	58.2%	56.5%	72.0%	58.2%	63.3%
KNN	63.0%	63.2%	62.1%	63.0%	62.7%
Naïve Bayes	60.1%	58.8%	67.5%	60.1%	62.8%
Random Forest	64.2%	66.1%	67.3%	64.4%	66.7%
XGBoost	63.0%	63.9%	70.9%	63.5%	67.2%

References

- Ağralı, Ö., & Aydın, Ö. (2021). Tweet Classification and Sentiment Analysis on Metaverse Related Messages. *Journal of Metaverse*, 1(1), 25–30.
- Al-Laith, A., Shahbaz, M., Alaskar, H. F., & Rehmat, A. (2021). Arasencorpus: A semi-supervised approach for sentiment annotation of a large arabic text corpus. *Applied Sciences (Switzerland)*, 11(5). <https://doi.org/10.3390/app11052434>
- Al-Shabi, M. A. (2020). Evaluating the performance of the most important Lexicons used to Sentiment analysis and opinions Mining. *IJCSNS International Journal of Computer Science and Network Security*, 20(1). <https://www.researchgate.net/publication/343473213>
- Anbazhagu, U. V., & Anandan, R. (2021). Predicting the Cricket Match Outcome Using ANFIS Classifier for Viewers Opinions on Twitter Data. *Data Management, Analytics and Innovation: Proceedings of ICDMAI 2020*, 1174, 171–182. https://doi.org/10.1007/978-981-15-5616-6_12
- Bailey, M., & Clarke, S. R. (2006). PREDICTING THE MATCH OUTCOME IN ONE DAY INTERNATIONAL CRICKET MATCHES, WHILE THE GAME IS IN PROGRESS. ©*Journal of Sports Science and Medicine*, 5(4), 480–487. <http://www.jssm.org>
- Batuwita, R., & Palade, V. (2010). Efficient Resampling Methods for Training Support Vector Machines with Imbalanced Datasets. *Efficient Resampling Methods for Training Support Vector Machines with Imbalanced Datasets in The 2010 International Joint Conference on Neural Networks (IJCNN)*, 1–8.
- Boateng, E. Y., Otoo, J., & Abaye, D. A. (2020). Basic Tenets of Classification Algorithms K-Nearest-Neighbor, Support Vector Machine, Random Forest and Neural Network: A Review. *Journal of Data Analysis and Information Processing*, 08(04), 341–357. <https://doi.org/10.4236/jdaip.2020.84020>

- Chan, J. Y. Le, Leow, S. M. H., Bea, K. T., Cheng, W. K., Phoong, S. W., Hong, Z. W., & Chen, Y. L. (2022). Mitigating the Multicollinearity Problem and Its Machine Learning Approach: A Review. In *Mathematics* (Vol. 10, Issue 8). MDPI. <https://doi.org/10.3390/math10081283>
- Charilaou, P., & Battat, R. (2022). Machine learning models and over-fitting considerations. *World Journal of Gastroenterology*, 28(5), 605–607. <https://doi.org/10.3748/wjg.v28.i5.605>
- Cheang, B., Wei, B., Kogan, D., Qiu, H., & Ahmed, M. (2020). *Language Representation Models for Fine-Grained Sentiment Classification*. <http://arxiv.org/abs/2005.13619>
- Chiny, M., Chihab, M., Chihab, Y., & Bencharef, O. (2021). LSTM, VADER and TF-IDF based Hybrid Sentiment Analysis Model. *International Journal of Advanced Computer Science and Applications*, 12(7), 265–275. <https://doi.org/10.14569/IJACSA.2021.0120730>
- Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.
- Friedman, J. H. (2001). GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE. *The Annals of Statistics*, 29(5), 1189–1232.
- Garg, A., & Tai, K. (2012). Comparison of regression analysis, Artificial Neural Network and genetic programming in Handling the multicollinearity problem. *International Conference on Modelling, Identification and Control*, 353–358. <https://dr.ntu.edu.sg/handle/10220/12899>
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media, Inc. <https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>

- Ghasiya, P., & Okamura, K. (2021). Investigating COVID-19 News across Four Nations: A Topic Modeling and Sentiment Analysis Approach. *IEEE Access*, 9, 36645–36656. <https://doi.org/10.1109/ACCESS.2021.3062875>
- Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019). A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification. *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, 24–28.
- Gie Yong, A., & Pearce, S. (2013). A Beginner's Guide to Factor Analysis: Focusing on Exploratory Factor Analysis. *Tutorials in Quantitative Methods for Psychology*, 9(2), 79–94.
- Godin, F., Zuallaert, J., & Vandersmissen, B. (2015). Beating the Bookmakers: Leveraging Statistics and Twitter Microposts for Predicting Soccer Results. *Proceedings of the 2014 KDD International Workshop on Large-Scale Sports Analytics*, 17–21.
- Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. *Journal of Big Data*, 7(1). <https://doi.org/10.1186/s40537-020-00305-w>
- Hatharasinghe, M. M., & Poravi, G. (2019). Data Mining and Machine Learning in Cricket Match Outcome Prediction: Missing Links. *2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019*, 1–4. <https://doi.org/10.1109/I2CT45611.2019.9033698>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- Hutto, C. J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media*, 216–225.
- Kaluarachchi, A., & Varde, A. S. (2010). CricAI: A classification based tool to predict the outcome in ODI cricket. *Proceedings of the 2010 5th International Conference on Information and Automation for Sustainability*,

ICIAfS 2010, December 2010, 250–255.
<https://doi.org/10.1109/ICIAFS.2010.5715668>

Kampakis, S., & Adamides, A. (2014). *Using Twitter to predict football outcomes*. <http://arxiv.org/abs/1411.1243>

Kampakis, S., & Thomas, W. (2015). *Using Machine Learning to Predict the Outcome of English County twenty over Cricket Matches*. 1–17.
<http://arxiv.org/abs/1511.05837>

Kannolli, B. S., Bevinmarad, P. R., Student, T., & Science, C. (2017). Analysis and Prediction of Sentiments for Cricket Tweets Using Hadoop. *International Research Journal of Engineering and Technology (IRJET)*, 4(10), 987–993.

Lamsal, R., & Choudhary, A. (2018). *Predicting Outcome of Indian Premier League (IPL) Matches Using Machine Learning*.
<http://arxiv.org/abs/1809.09813>

Lay, V., Lee, S., Gan, K. H., Tan, T. P., Abdullah, R., Lay, V., Lee, S., Gan, K. H., Tan, T. P., & Abdullah, R. (2019). Semi-supervised Learning for Sentiment Classification using Small Number of Labeled Data. *Procedia Computer Science*, 161, 577–584.

Liang, H., Sun, X., Sun, Y., & Gao, Y. (2017). Text feature extraction based on deep learning: a review. In *Eurasip Journal on Wireless Communications and Networking* (Vol. 2017, Issue 1). Springer International Publishing.
<https://doi.org/10.1186/s13638-017-0993-1>

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. <http://arxiv.org/abs/1907.11692>

Loureiro, D., Barbieri, F., Neves, L., Anke, L. E., & Camacho-Collados, J. (2022). *TimeLMs: Diachronic Language Models from Twitter*.
<http://arxiv.org/abs/2202.03829>

Mohsin Abdulazeez, A. (2021). Impact of Deep Learning on Transfer Learning : A Review Deep Learning View project Gait recognition with wavelet

- transform View project. *International Journal of Science and Business*, 5(3), 2014–2216. <https://doi.org/10.5281/zenodo.4559668>
- Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.
- Phulare, P., & Deshmukh, S. N. (2021). Cricket Twitter Data Sentiment Analysis and Prediction Exerted Machine Learning. *Proceedings of IEEE International Conference on Disruptive Technologies for 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications*, 141–145. <https://doi.org/10.1109/CENTCON52345.2021.9688197>
- Pramanik, M. A., Suzan, M. M. H., Biswas, A. A., Rahman, M. Z., & Kalaiarasi, A. (2022). Performance Analysis of Classification Algorithms for Outcome Prediction of T20 Cricket Tournament Matches. *2022 International Conference on Computer Communication and Informatics, ICCCI 2022*. <https://doi.org/10.1109/ICCCI54379.2022.9740867>
- Salih Hasan, B. M., & Abdulazeez, A. M. (2021). A Review of Principal Component Analysis Algorithm for Dimensionality Reduction. *Journal of Soft Computing and Data Mining*, 02(01). <https://doi.org/10.30880/jscdm.2021.02.01.003>
- Sankaranarayanan, V. V., Sattar, J., & Lakshmanan, L. V. S. (2014). Auto-play: A data mining approach to ODI cricket simulation and prediction. *SIAM International Conference on Data Mining 2014, SDM 2014*, 2, 1064–1072. <https://doi.org/10.1137/1.9781611973440.121>
- Schumaker, R. P., Jarmoszko, A. T., & Labedz, C. S. (2016). Predicting wins and spread in the Premier League using a sentiment analysis of twitter. *Decision Support Systems*, 88, 76–84. <https://doi.org/10.1016/j.dss.2016.05.010>
- Sharma, S. (1995). *Applied Multivariate Techniques*. Wiley & Sons, Inc..
- Singhvi, A., Shenoy, A. V, Racha, S., & Tunuguntla, S. (n.d.). *Prediction of the outcome of a Twenty-20 Cricket Match*.

- Swartz, T. B. (2017). Research Directions in Cricket. In *Handbook of statistical methods and analyses in sports* (pp. 461–476). Chapman and Hall/CRC.
- Ul Mustafa, R., Nawaz, M. S., Lali, M. I. U., Zia, T., & Mehmood, W. (2017). Predicting the Cricket match outcome using crowd opinions on social networks: A comparative study of machine learning methods. *Malaysian Journal of Computer Science*, 30(1), 63–76. <https://doi.org/10.22452/mjcs.vol30no1.5>
- Wickramasinghe, A. N., & Yapa, R. D. (2018). Cricket match outcome prediction using tweets and prediction of the man of the match using social network analysis: Case study using IPL data. *18th International Conference on Advances in ICT for Emerging Regions, ICTer 2018 - Proceedings, 2014*, 1–1. <https://doi.org/10.1109/ICTER.8615563>
- Wickramasinghe, I. (2022). Applications of Machine Learning in cricket: A systematic review. *Machine Learning with Applications*, 10, 100435. <https://doi.org/10.1016/j.mlwa.2022.100435>
- Yang, L., & Shami, A. (2020). On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *Neurocomputing* 415, 295–316. <https://doi.org/10.1016/j.neucom.2020.07.061>
- Zhao, L., Li, L., & Zheng, X. (2021). A BERT based Sentiment Analysis and Key Entity Detection Approach for Online Financial Texts. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 1233–1238.

Internet Sources

cric-tweets-sentiment-analysis. (2022, December). <https://huggingface.co/>. Retrieved March 9, 2023, from <https://huggingface.co/sppm/cric-tweets-sentiment-analysis>

DIGITAL 2023: GLOBAL OVERVIEW REPORT. (2023). In <https://datareportal.com/>. Retrieved March 9, 2023, from <https://datareportal.com/reports/digital-2023-global-overview-report>

ICC Men's Twenty20 International Playing Conditions. (2022). In <https://www.icc-cricket.com/>. Retrieved March 9, 2023, from <https://resources.pulse.icc-cricket.com/ICC/document/2022/10/13/28311034-6d66-436e-a20a-6d9eaf1d2c24/ICC-Men-s-T20I-Playing-Conditions-October-2022.pdf>

Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing*. Draft.

Phil Lynch on Manchester United's media strategy and "the Ronaldo effect." (2021, October 27). Retrieved March 9, 2023, from <https://podcasts.google.com/feed/aHR0cHM6Ly9hdWRpb2Jvb20uY29tL2NoYW5uZWxzLzUwNjMyMDIucnNz/episode/dGFnOmF1ZGlvYm9vbS5jb20sMjAyMS0xMC0yNjovcG9zdHMuNzk2NzQ5MQ%3D%3D>

Tam, A. (2021, October 20). Principal Component Analysis for Visualization. *Machine Learning Mastery*. Retrieved March 9, 2023, from <https://machinelearningmastery.com/principal-component-analysis-for-visualization/>

Understanding LSTM Networks. (2015, August 27). Retrieved March 8, 2023, from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>