University of Cape Town

Modelling First Innings Totals in T20 Cricket: Applications in the Indian Premier League

Student:
Arlton W Gilbert
GLBARL001

Supervisor:
Mr Stefan S Britz

Minor Dissertation submitted in partial fulfilment of the requirements for the degree of Master's in Data Science at the

Department of Statistical Sciences

October 11, 2023

Plagiarism Declaration

I, Arlton Wayne Gilbert, declare that this thesis titled, 'Modelling First Innings Totals in T20 Cricket: Applications in the Indian Premier League' and the work presented in it are my own. I confirm that:

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is my own.
- I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own.

Signed: Signed by candidate

Date: 11-October-2023

UNIVERSITY OF CAPE TOWN

Abstract

Modelling First Innings Totals in T20 Cricket: Applications in the Indian Premier League

- by Arlton Wayne Gilbert

In the game of cricket, teams batting first are faced with the question of how many runs are enough. This paper proposes a solution to this in the context of the Indian Premier League (IPL). The aim is to build a model that will allow teams to determine what scores they would need to score for any given confidence of avoiding defeat in regular time, viz. before any Super Overs.

The following machine learning methods are considered for this purpose: logistic regression, classification trees, bagging, random forest, boosting, support vector machines, artificial neural networks, and naive Bayes. Features are chosen that represent various key aspects of the game, including player strengths, stadium information, the winner of the toss, and which teams are involved.

The results show that logistic regression is the best performing model, having a prediction accuracy of 70.27% and a Brier score of 0.2 for the 2022 season of the IPL. The majority of the incorrect predictions occurred in prediction ranges where the model itself suggested the game could have gone either way.

The model is, therefore, fit for purpose and can allow teams to pace their innings and reduce unnecessary risks. The model can also be trained and used on other limited-over tournaments, including one-day matches.

Acknowledgements

I would like to offer my sincere thanks to the following people:

- Stefan Britz - my supervisor: who was always there to help on short notice for the rather long time this thesis took.
- Megan Gilbert - my partner: who, over the course of this paper went from being my girlfriend to my fiancée to my wife, and has always been there to support me.
- Joshua and Indira Gilbert - my parents: who instilled in me from a young age the importance of education.

- Arlton Gilbert, February 2023

Contents

List of Figures

List of Tables

Acronyms

ANN	**A**rtificial **N**eural **N**etworks
CSK	**C**hennai **S**uper **K**ings
CV	**C**ross-**v**alidation
DC	**D**elhi **C**apitals
DLS	**D**uckworth-**L**ewis-**S**tern
D/N	**D**ay/**N**ight
GL	**G**ujarat **L**ions
GT	**G**ujarat **T**itans
HDC	**H**yderabad **D**eccan **C**hargers
IPL	**I**ndian **P**remier **L**eague
KKR	**K**olkata **K**night **R**iders
KTK	**K**ochi **T**uskers **K**erala
k-NN	**k**-**N**earest **N**eighbours
LSG	**L**ucknow **S**uper **G**iants
MI	**M**umbai **I**ndians
MSE	**M**ean **S**quare **E**rror
NRR	**N**et **R**un **R**ate
ODI	**O**ne **D**ay **I**nternational
OOB	**O**ut-**o**f-**b**ag
PBKS	**P**unja**b** **K**ing**s**
PDP	**P**artial **D**ependence **P**lot
PW	**P**une **W**arriors
RCB	**R**oyal **C**hallengers **B**angalore
ROC	**R**eceiver **O**perating **C**haracteristic
RPS	**R**ising **P**une **S**upergiant
RR	**R**ajasthan **R**oyals
SRH	**S**un**r**isers **H**yderabad
SVM	**S**upport **V**ector **M**achine
T20	**T**wenty**20**
UAE	**U**nited **A**rab **E**mirates
WASP	**W**in **a**nd **S**core **P**redictor

Chapter 1

Introduction

This study aims to model the predicted probabilities of victory for a team at the halfway stage of a 20-over game of cricket. If successful, this will allow teams to aim for specific targets in the first innings in order to have a certain confidence of victory. A background to the problem is provided in this chapter in order to understand the need for such a study. The background includes brief discussions of the game of cricket, the rise of the Indian Premier League (IPL), sports predictions and machine learning. Thereafter, the specific aims of this paper are set out by looking at the research problem and research objectives. Finally, an outline of the paper is given.

1.1 Background

In limited-overs cricket, the team batting first is faced with the question of how many runs they need to score. Using a range of factors available before the game, this paper explores potential models that can be used to determine what scores a team batting first would have to score for a given confidence of victory. The models explored will make use of specific characteristics of the teams involved, such as a measure of the batter's and bowler's strengths. Although the models should, in theory, work for all limited-overs cricket formats, this paper will be based on the 20-over T20 format and, in particular, the Indian Premier League (IPL), owing to data availability.

1.2 Cricket

Cricket is the second most popular sport in the world behind soccer (Jhawar and Pudi, 2016). Before delving into the modelling, it is important to have a basic understanding of the game of cricket and the IPL.

1.2.1 Basics of Cricket

Cricket is a bat-and-ball game played between two teams of 11 players each. The field is oval shaped, with a rectangular 22-yard-long pitch at the centre. Teams take turns to bat and score as many runs as possible, which their opponents subsequently try to better. When batting, a team attempts to score runs by hitting a ball bowled at them and then running between the wickets. A full length run between the wickets equates to one run. A team can score extra runs by hitting the ball to the boundary of the field, with four runs being scored if the ball bounces at least before reaching the boundary and six runs if the ball goes over the boundary without bouncing.

The team which is batting is known as the *batting team* and the other team is known as the *fielding* or *bowling team*. The fielding team attempts to prevent the batting team from scoring runs. Batters can also be dismissed by various methods, the most common of these being:

- **Bowled:** If a bowler delivers a ball that hits the stumps.
- **Caught:** If a batter hits a ball and a fielder catches it before it touches the ground.
- **Leg Before Wicket:** If the ball would have hit the wicket but is intercepted by any part of the batter except their bat or gloves.
- **Run Out:** If the fielding team hits the wicket with the ball while a batter is outside their crease after attempting a run.
- **Stumped:** If the fielder positioned directly behind the batter's stumps, known as the wicketkeeper, hits the wicket with the ball while the batter is outside their crease directly after a ball is bowled.

Each team bats either once or twice, depending on the format. At the end of the match, the team with the most runs wins. Each completed round of batting for a team is known as an *innings*. Each delivery or single round of play between a batter and bowler is known as a *ball*. An *over* comprises six legal balls. Currently, there are three main formats for cricket:

- **Test Cricket:** Matches are played over five consecutive days, lasting about seven and a half hours per day with a few breaks in between. Teams bat two innings each with no limit on the number of overs per team. If both teams have not been dismissed at the end of five days, the match is a draw.
- **One-Day:** Teams bat one innings each with an innings comprising 50 overs. Matches typically last approximately eight hours.
- **Twenty20 (T20):** This is the shortest format of the game. Teams bat one innings each with an innings comprising 20 overs. Matches typically last approximately three hours.

The latter two formats here are known as limited overs cricket due to restrictions on the total number of overs that a team can face.

1.2.2 The Rise of the Indian Premier League (IPL)

Smith (2021) explained that in response to a fall in attendance and a lack of sponsorship revenue in the early 2000s, English cricket needed a fresh and exciting domestic competition. The proposed shortened version of the game, T20 cricket, would be closer to the timespan of other popular team sports, and it was hoped that the fast-paced game would attract more spectators at the ground and more viewers on television. The first T20 tournament, *The Twenty20 Cup*, was launched in 2003 in England. Attendance at matches was incredible, and the tournament was deemed a huge success.

Other countries soon took to the new format. However, as reported by The Hindu (2019), the Indian cricket board was initially against the idea of this shortened version of the game and almost did not send a team to the first T20 World Cup in 2007. At the last moment, India decided to join and ended up winning the inaugural T20 World Cup. A year later, the Indian Premier League (IPL), India's first domestic T20 league, was launched to great fanfare.

Financial backing for the tournament was substantial, allowing the participating teams to attract the best cricketers from all around the world. The IPL allowed each team to field up to four international cricketers in their matchday teams. The tournament has been enormously successful, with regularly sold-out stadiums and large international viewership. Laghate (2020) reported that almost half of the Indian television viewership watched the IPL in 2020. Star India, the global broadcaster of the IPL, reportedly earned close to 27 billion Indian Rupees (around R5.8 billion) in advertising revenue for the tournament.

The following are seen as some of the primary reasons that the IPL has become so successful: pre-existing popularity of cricket, star power of players involved, celebrity team owners, short format, strategic marketing, big sponsorships and extensive coverage worldwide.

1.2.3 Format of the IPL

The number of teams has varied over the years from eight to 10 teams. In all seasons, except 2011 and the most recent season, 2022, the initial stage was a traditional round-robin format where all teams played each other both home and away, once each. In the 2022 season, two new franchises were introduced, and the format changed slightly to one similar to the format used in the 2011 season.

Gollapudi (2022) detailed how the format for the 2022 season worked. For the 2022 season, 10 teams were seeded based on the number of IPLs they have won, or finals they were in in previous seasons. The teams were then placed into two "virtual" groups, with teams ranked 1, 3, 5 and 7 in Group A and teams ranked 2, 4, 6 and 8 in Group B. The two new franchises were randomly allocated to one of these two groups. These virtual groups can be seen in Figure 1.1. Teams played all other teams in the same virtual group or row as them twice each, and the four other teams once each. Mumbai Indians (MI), for example, played the four other teams in their group - Kolkata Knight Riders (KKR), Rajasthan Royals (RR), Deccan Capitals (DC) and Lucknow Super Giants (LSG) - twice each, the other team in their row - Chennai Super Kings (CSK) - twice each, and the other four teams - Sunrisers Hyderabad (SRH), Royal Challengers Bangalore (RCB), Punjab Kings (PBKS) and Gujarat Titans (GT) - once each. All teams, therefore, played a total of 14 matches in this initial phase. These groups were called *virtual* because teams were still ranked based on points as if they were part of one group.

Figure 1.1: Group format for the 2022 IPL

Source: Gollapudi (2022)

The 2022 season also did not follow the usual home and away format. Gollapudi (2022) explained that due to complications caused by the COVID-19 pandemic, there were only four venues across two cities, Mumbai and Pune, with three venues used in the former and one venue used in the latter. Figure 1.2 shows the number of matches played at each venue. Every team played four times each at the Wankhede Stadium and DY Patil Sports Academy and three times each at the Brabourne Stadium and the MCA International Stadium.

Figure 1.2: Breakdown of the 2022 IPL stadiums used

Source: Gollapudi (2022)

In all seasons, the top four teams advanced to the playoff stage. If two teams have the same points, the team with a higher net run rate (NRR) will rank higher. NRR measures how many more runs a team scores per over than they concede. The first three seasons of the IPL featured a traditional semi-final and final, with the third season having a third and fourth place playoff between the losing semi-finalists. Since the fourth season (2011), the format in Figure 1.3 has been used. This gives the teams that finish first and second an additional advantage by allowing them a shorter route to the final as well as a second chance if they lose the first match.

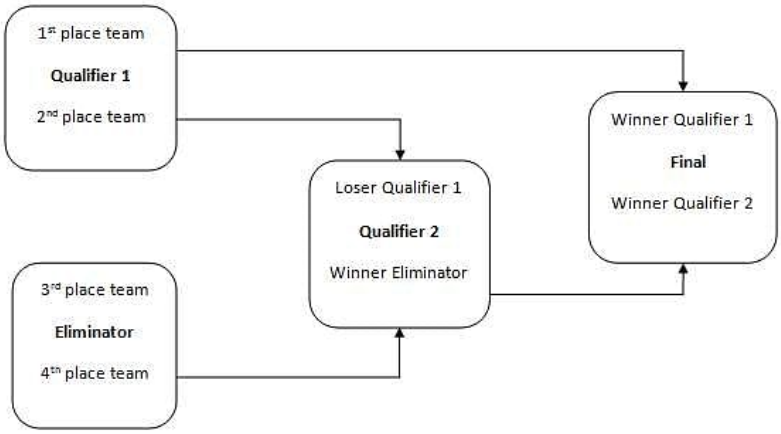


Figure 1.3: Format of the IPL playoffs

Source: Khare (2020)

All ties in the IPL are settled by the *Super Over*, where each team bats for one over each, with the winner being the team which has scored more runs in that over. Each team is allowed to select only three batters to represent them for this particular over. If a team loses two wickets before the over is completed, their innings is over. The Super Over is repeated until there is a winner. The period of the match before the Super Over will be referred to as *regular time* in this paper.

According to First Post (2022), prize money for the winner of the 2022 IPL season was 200 million Indian Rupees (approximately R42.1m) and for the runner-up 125 million Indian Rupees (approximately R26.3m). The margins for victory are small, with matches often coming down to the last over and sometimes even the Super Over.

1.3 Sports Predictions

Statistical modelling has been used in sports for decades to derive valuable information from games and to predict possible outcomes. As well as providing insight to the teams involved, there is a huge betting market and enormous media coverage, which have given strong incentives to model the game from a range of perspectives (Nimmagadda et al., 2018). For teams, strategies can be developed that take into account specifics about the opponent and other historical information about the team itself, the venue, etc. (Bunker and Thabtah, 2019).

The online gambling industry is rapidly growing, with a $20 billion (around R358 billion) online gambling market in 2009 becoming a $40 billion (around R716 billion) industry in 2016, of which about 40% was sports betting (Kapadia et al., 2019). This incentivises betting houses to develop models that aid them in determining odds and for users to develop models that give them additional insight into games.

The 2011 film *Moneyball*, which received six Academy Award nominations, brought the topic of sports predictions using statistics to the public eye, using the sport of baseball. The movie was based on the 2004 book of the same name where author Lewis (2004) introduced a new analytic system, dubbed Sabermetrics, which looked for creative, objective ways to quantify a player's ability to help his team win games. The system looked far beyond the traditional batting averages and runs batted in. Players are analysed with an extensive list of variables, including VORP (value over replacement player), BABIP (batting average on balls in play), DIPS (defense-independent pitching statistics), runs created, range factor, and the Pythagorean winning percentage, amongst others (Reider, 2014).

1.4 Machine Learning

Machine learning is defined as a process of programming computers to use example data or historical information (Alpaydin, 2020). It involves using algorithms to find patterns and make predictions or decisions based on input data. Machine learning is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from a given context (El Naqa and Murphy, 2015). It is one of the most popular approaches used in sports analytics research.

There are various types of machine learning algorithms. Of interest here will be supervised learning models, which allow one to feed in a range of independent variables and then make a prediction of a final dependent variable. The model learns from historical data, which it then uses to determine the most likely outcome given a particular set of inputs. The main classes in machine learning are presented in Figure 1.4.

For this purpose, the category of interest will be *classification*. Classification is the part of supervised learning that focuses on models where the outcomes are discrete or fit into categories.

1.5 Research Problem

Cricket may be known as a battle between bat and ball, but it is so much more than that. In every match, a range of factors influence the outcome. The question of how many runs are

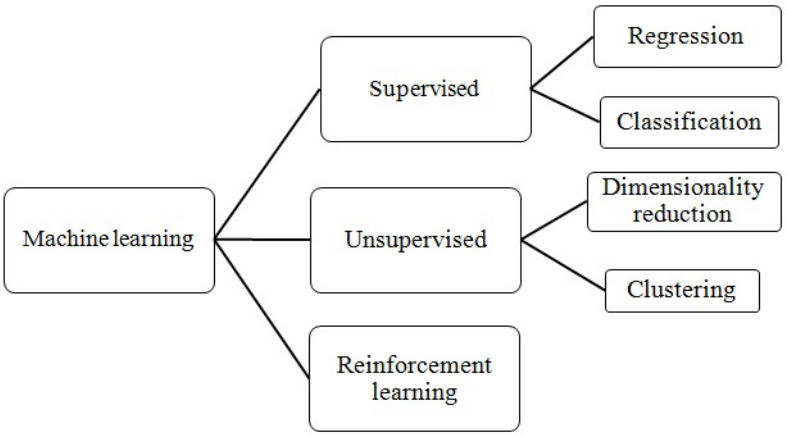


Figure 1.4: Breakdown of machine learning categories

Source: Kumar, Shahani, and Karchi (2020)

enough to defend for the team batting first is, therefore, far from a simple one. T20 cricket has relatively high run rates (runs per over) compared to the other formats. However, higher run rates often come with higher risk-taking. Trying to score too quickly often comes with unnecessary risks. Teams tend to lose too many wickets and usually end up with much lower totals.

If teams had a model that could inform them how many runs are needed for a certain confidence of victory given all the factors at hand, they could use this to pace their innings. During an innings, teams could also determine that the extra confidence of victory secured by scoring a certain number of extra runs may not be worth the risks in trying to obtain those runs.

1.6 Research Objectives

The main research objective is to develop a model that can provide first innings total targets for different confidence levels of victory. This can be used by teams to pace their innings. The targets will not change as the match progresses as they are total scores to set for the team batting second. The model will use the specific strengths of the players involved as well as a range of other factors available at the start of the match, such as stadium information and toss winner.

The following machine learning models will be considered: logistic regression, classification trees, bagging, random forest, boosting, support vector machines (SVMs), artificial neural networks (ANNs) and naive Bayes. Models will be compared on their test accuracy and Brier score, which is a measure of the mean squared difference between the predicted probability and the actual outcome (see Section 4.1.3). The performance of models will also be examined at different prediction probability ranges to see how well they perform in different ranges. For example, suppose a model predicts a 60% probability of victory for the team batting first. In that case, only six out of 10 matches that fall in that category ending with a victory for the team batting first is consistent with the model's predictions.

In order to achieve the main research objective, the following sub-objectives need to be met:

1. Understand the current research around sports predictions.
    (a) Identify models in use for sports predictions.

(b) Identify factors in use for cricket predictions.

2. Understand the data.

   (a) Find adequate sources of data.

   (b) Define predictor variables and transform data to fit.

   (c) Understand data structure, trends and anomalies.

3. Develop machine learning models for modelling of first innings totals in the IPL.

   (a) Define the methodology.

   (b) Explain the models used.

   (c) Fit and optimise models.

   (d) Evaluate results and choose the best model for purpose.

1.7 Chapter Outline

An overview of the dissertation is provided below, which details what is contained in each chapter. Table 1.1 then sets out the research objectives addressed in each chapter.

Chapter 1: Introduction
In this chapter, the background to the problem is given. The need for such a paper is outlined, and the research objectives are set.

Chapter 2: Literature Review
In this chapter, literature related to the problem at hand is explored. Previous research related to the topic is summarised and analysed, and possible shortcomings of these are highlighted.

Chapter 3: Data
In this chapter, the data used and the features extracted from the data are explained. Exploratory data analysis is also covered here.

Chapter 4: Methodology
In this chapter, the methodology used is explained. Details of the models used are provided, along with explanations of how hyperparameters are chosen and parameters are tuned.

Chapter 5: Model Application
In this chapter, the results for each of the models used are presented, and a comparison of these models is performed with the purpose of choosing the best model for the aim of this paper.

Chapter 6: Conclusion
In this chapter, a summary of this paper is provided with a particular focus on the results and key takeaways. Finally, possible next steps that can be done in future work are suggested.

Chapter	Title	Sub-objectives Covered
2	Literature Review	1a, 1b
3	Data	2a, 2b, 2c
4	Methodology	3a, 3b
5	Model Application	3c, 3d

Table 1.1: Chapter outline with research sub-objects covered

Chapter 2

Literature Review

In this chapter, available research on the topic under paper is reviewed. It begins with a look at machine learning in sports historically. Next, some of the research on score predictions in cricket is discussed. Finally, factors shown to have possible effects on cricket predictions from previous studies are explored.

2.1 Machine Learning in Sport

Machine learning has many applications in sports. Horvat and Job (2020) explained that machine learning algorithms aim to assist coaches and managers not only predict the outcome of games but also to assess a player's or team's performance, to identify possible injuries, and to assist with scouting. It can also be used to make sport-betting decisions.

Horvat and Job (2020) attempted to analyse some of the existing machine learning algorithms for predicting sports outcomes. They analysed over 100 existing papers on the topic, with 39 of them then discussed more thoroughly in their paper. Figure 2.1 breaks down the number of papers using a particular machine learning algorithm. Neural networks were by far the most popular algorithm.

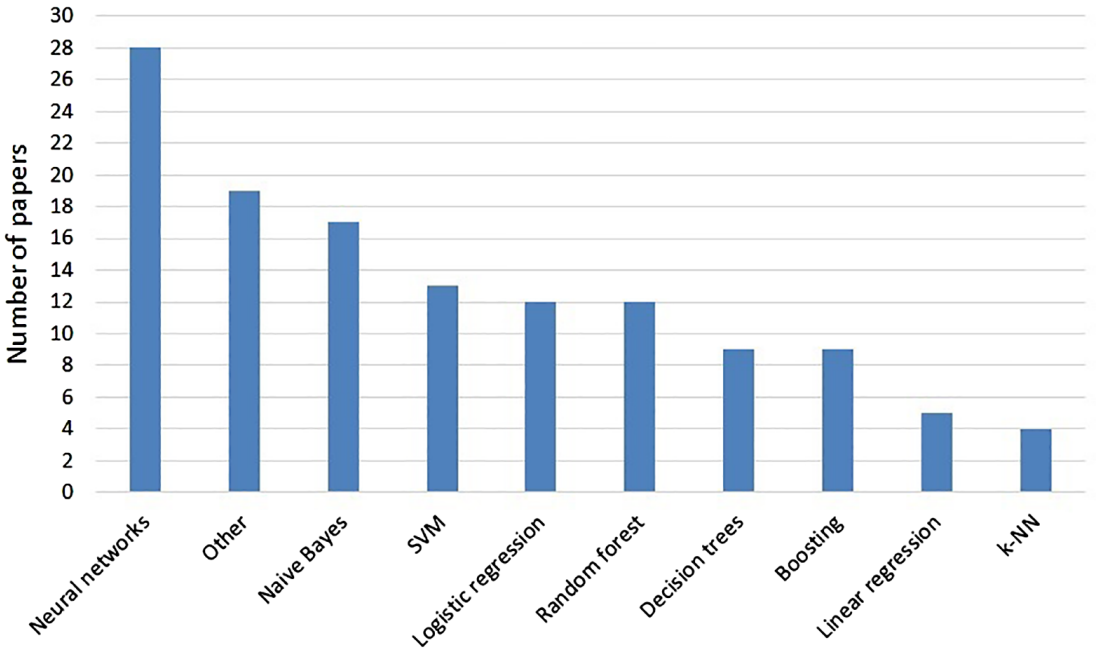


Figure 2.1: Number of papers analysed by Horvat and Job (2020) using a particular machine learning algorithm

Source: Horvat and Job (2020)

Figure 2.2 breaks down the prediction accuracy for the papers analysed by Horvat and Job (2020) with the accuracy for the cricket papers highlighted.

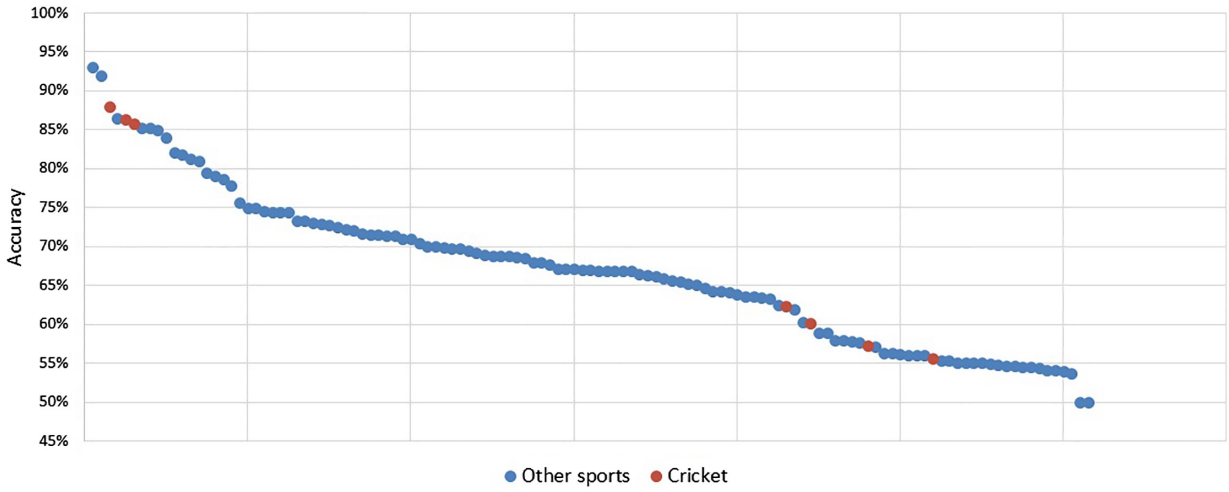


Figure 2.2: Cricket prediction results in relation to other analysed sports

Source: Horvat and Job (2020)

Of particular note is the prediction accuracies of Mustafa et al. (2017). The authors used sentiment analysis on Twitter by collecting the total number of tweets before a match for each team, fans' sentiment towards each team and fan score predictions. They achieved a result of 87.9% using the SVM method, 86.28% using naive Bayes and 85.73% using logistic regression for the *2015 Cricket World Cup*. This paper will not make use of any sentiment analysis.

The other cricket results in Figure 2.2 that are sourced from the paper by Kampakis and Thomas (2015) are more in line with the methods this paper will use and will be discussed in the next section.

2.2 Score Prediction in Cricket

Over the years, many people have proposed models to predict final scores in cricket at various points in the match. The most famous of these models is the Duckworth-Lewis-Stern (DLS) method presented by Duckworth and Lewis (1998). Initially introduced in 1997 as the Duckworth-Lewis method, Stern (2016) introduced modifications to the methodology to better fit modern trends in cricket. DLS uses the concept of resources based on how many wickets have fallen and how many overs have been faced in order to predict the final score. A mathematical formula is used to calculate the resources remaining table using these two factors. This table is then used to scale up the current score based on how much of their resources teams have left.

The DLS method, however, uses predictions based on generic teams at that point in time and does not consider any of the specifics of the teams involved. This makes sense for this particular system as it is actually used to adjust scores in matches where playing time is reduced significantly for any reason (usually rain). It would not be fair on weaker teams if stronger teams were automatically given an advantage by adjusting scores more favourably for stronger teams.

Recently, another method known as the Win and Score Predictor (WASP) has become popular on broadcasting stations, especially in New Zealand, to show live win probabilities. For the team batting first, it predicts the final total at any point in the match; for the team

batting second, it gives the likelihood of the chasing team winning. Since this is a propriety model, very little has been written about the finer workings of the model. New Zealand Cricket (2014) explained that WASP does try to account for the history of matches at a venue and conditions on the day by having an expert setting a "par score" before play starts. Again, this leaves something up to interpretation, whereas a more objective model would be preferred here. WASP also does not consider the actual teams involved.

Kapadia et al. (2019) worked on predicting the outcome of IPL matches before the matches have started using two sets of data - one pertaining to the home team advantage and the other on toss decision. They fitted a variety of models and attained an accuracy of 57% using naive Bayes on the home team set and 62% using k-nearest neighbours (k-NN) for the toss winner set. They also did not use any player-specific factors.

Kampakis and Thomas (2015) also published a cricket prediction paper which was analysed by Horvat and Job (2020) as seen in Figure 2.2. They performed an investigation for English county cricket 20-over matches that incorporated the players' strengths. They used different combinations of common statistics to get their player strengths. They predicted the winner in 62.4% of matches using naive Bayes, 60.1% for logistic regression, 57.2% for decision tree and 55.6% for random forest. They pointed to the greater unpredictability of 20-over cricket, leading to lower percentages than that observed in many other sports. They pointed out that this was an improvement on the levels present in the gambling industry at the time.

Jhawar and Pudi (2016) used machine learning techniques to predict the outcome of one-day internationals (ODIs) from 2010 to 2014. They used three features in their model: toss, venue and relative team strength. Relative team strength was derived from the overall bowling and batting strengths of the teams involved. These strengths were derived using both the complete history and recent history to measure form. They combined player strengths by looking at how likely that player was to perform that action; e.g. for batting, they looked at how often each batter actually batted in matches they played. They produced a k-NN model with 71% accuracy.

Viswanadha et al. (2017) built on the work by Jhawar and Pudi (2016) to form predictions for the winner at the end of each over in the second innings of an IPL match. They used measures to incorporate the dynamically updating match context as well as the relative strengths of the players who still had a direct batting or bowling role to play in the match. Using random forest, they achieved an accuracy of 65.79% - 84.15% over the course of the second innings.

Tekade et al. (2020) also sought to predict the winner before the start of an IPL match. They attained an impressive accuracy of 90%, which is far greater than similar papers in the field. They did not use any player-specific information but instead used the teams' batting averages, bowling economy, and strike rates. They also used some following unique features related to the conditions at the time of the match:

- Pitch conditions: Whether it was a batting pitch, bowling pitch or neither.
- Temperature: Maximum and minimum temperatures.
- Humidity: A measure of the water vapour level in the air.
- Precipitation: a measure of the percentage chance of rain.

The justification for each of these conditions will be discussed in the next section, where the choice of some of the features used in the models is justified.

2.3 Feature Discussion

Gómez-Ruano, Pollard, and Lago-Peñas (2021) looked at the research on home-ground advantage in cricket and concluded that there is evidence of a home-ground advantage effect. However, according to the authors, the exact factors contributing to this advantage and the magnitude of influence that each of these factors have are not clear. They noted that there has been limited empirical investigation of the four major proposed factors:

- The influence of the home crowd on player performance and umpire bias.
- The physiological effects of travelling on opposition players.
- Rules such as privileges favouring the home team.
- Familiarity with the physical environment.

Silva and Swartz (1998), in their investigation of ODI cricket matches played during the 1990s, concluded that home-ground advantage increases the log-odds of the probability of winning by approximately 0.5 and that, contrary to popular belief, winning the coin toss provides no competitive advantage. Kaluarachchi and Varde (2010) reached a similar conclusion regarding the coin toss in their investigation into all ODI matches since 1971. Bandulasiri (2008), however, looked into ODI matches from the start of 1995 until the end of the *2007 Cricket World Cup* and concluded that the time of day affects the importance of the coin toss, with the toss being significant for 'day/night' matches. This paper will make use of coin toss for the following reasons: significance is still placed by the media on winning the coin toss, it is a simple addition to the features, and the papers mentioned here that disputed the significance of the coin toss were for a different form of the game. The time of day in terms of whether it is a 'day' or 'day/night' match will also be added as a feature.

Unlike many other sports, the shapes and sizes of cricket fields are not fixed except for the inner circle and pitch, which are 30 yards (radius from the stumps) and 22 yards (in length), respectively (Kumar et al., 2021). This can have a significant impact on scores as players have to pay particular attention to any smaller or larger than normal boundaries. Ground name was one of the factors used by Kumar et al. (2021) in their model, which was able to predict the winner at varying stages of the match on average 71.08% of the time using random forest for ODI matches from 2008 to 2016.

Jhawar and Pudi (2016) and Viswanadha et al. (2017) used measures for the player strengths to great success, as mentioned in Section 2.2. The measures in these papers will form the basis for the player strength measures used in this paper.

Sentiment analysis similar to that used by Mustafa et al. (2017), which was detailed in Section 2.2 was considered. However, this removes the objectivity of the model as it leaves it open to some manipulation by, for example, mass tweeting by bots. Furthermore, the coding needed to extract this information would be a major project on its own and is, therefore, beyond the scope of this paper.

The next set of factors considered were those surrounding the conditions used in the paper by Tekade et al. (2020). These factors contributed to very high prediction accuracy in the paper and have very logical explanations for why they would have an impact. Before looking at these reasons, a few simple bowling concepts need to be defined:

- Spin: A bowling style where the bowler uses their grip and bowling action to impart rotation on the ball. This causes deviation from a straight path, making it difficult for a batter to predict the trajectory.
- Swing: The movement of the ball in the air after it is bowled.

- Seam: The movement of the ball after it hits the pitch.

Based on the explanations given by Tekade et al. (2020), the reasons those conditions could have an impact are as follows:

- Pitch conditions: A dry pitch is good for spinners. A wet pitch will have no swing and spin. A green pitch is helpful for swing bowlers.
- Temperature: Swing and seam depend on the temperature; the ball will swing more in cold weather.
- Humidity: In high humidity, the ball becomes heavier and loses its bounce. It may also swing less in high humidity.
- Precipitation: The pitch may become softer and slower. The ball may become harder to grip for bowlers, especially spinners.

These factors would undoubtedly have been helpful in this paper. However, this data does not appear to be available on any public database. All four authors were contacted via email in order to assist with the sourcing of this data. Unfortunately, none of the authors responded, and thus, these factors will be left out. Future studies would do well to communicate further with the authors on how they managed to produce accuracies so much higher than those produced in other studies, even those on one-day cricket.

2.4 Conclusion

In this chapter, the current research in fields related to this paper was explored. It started with an investigation into machine learning in sports in general, which showed that the topic was quite popular in sports. Models commonly used in the field and the results they produced for cricket compared to other sports were detailed. Attention was given to papers that produced high prediction accuracies or added something innovative to the field. Finally, consideration was given to what factors could be used in this paper by looking at the success of different factors in previous studies.

This chapter formed the basis from which the actual analysis in this paper proceeds. In Chapter 3, the data used is investigated, and an explanation of the variables derived from it is provided. In Chapter 4, the methodology is detailed which includes looking at how the data is split, the models used and the evaluation criteria. In Chapter 5, these models are fit, and the analysis is performed. Finally, in Chapter 6, there is a review of all the analyses, and the results are compared to the papers discussed in this chapter.

Chapter 3

Data

In this chapter, the data used in this paper is detailed. It begins with a background of the raw data and outlines how this was sourced. This is followed by a description of the features that were extracted from the data with further detail pertaining to how these features were extracted. Finally, exploratory data analysis is covered.

3.1 Background

The data used can be accessed from the *Cricket Commentary IPL - Cricbuzz dataset* on Kaggle. The file "IPL_SCHEDULE_2008_2020.csv" contains links to the match details of every IPL match played from the beginning of the IPL in 2008 until 2020. The links for the 2021 and 2022 matches were added using a web-scraper. These links were then used to scrape the scorecards for every single match. From these, the features were extracted. All coding for this was done in the programming language *R*.

In total, there were 958 IPL matches from the 2008 season until and including the 2022 season. Some features require a certain number of historical matches, and some will tend to be quite volatile over a short number of matches. The first two seasons, therefore, form part of the calculation of features but are not part of the observations used and will be referred to as the *burn-in* part of the data. The remaining observations were then broken down into a training and test set, with the 2022 season used for the test set.

Only those matches that were completed without the use of DLS will be included in the observations. However, if at least part of the match was played, the completed parts was used in the calculation of the various features. The breakdown of the 958 matches can be seen in Table 3.1.

Result	Burn-In	Training	Test	Total
Abandoned Before Ball Bowled	3	5	0	**8**
Abandoned During Match	0	4	0	**4**
Match Completed Using DLS	5	14	0	**19**
Match Tied and Super Overs Needed	1	13	0	**14**
Other Matches	109	730	74	**913**
Total Matches	**118**	**766**	**74**	**958**

Table 3.1: Results of all matches in the IPL 2008 - 2022

In Table 3.1, the first three rows were left out entirely from the final training and test data sets. The tied match case in the fourth row is interesting because it is neither a win nor a loss for the defending team, and classification could go either way. Given that not many matches were played, especially for some of the more complex models, it is best to avoid leaving out too many matches where possible. In this paper, it will be framed as a positive result for the

defending team, and the calculated probability will then be what score is enough to avoid a loss with a certain degree of confidence. The last two rows here, therefore, together make up the full set of observations once the first two seasons are excluded. Thus, the training and test dataset have 743 observations and 74 observations, respectively.

Note that in Table 3.1, there were no overlaps of categories. This may have occurred if there was a match that used DLS that was also tied before the Super Over or a match that was abandoned after scores were tied in regular time but before the Super Overs were completed. In the former case, the match would have been left out of the observations because it is a reduced overs match, and therefore, the result will not be used in this paper, and in the latter, it would be used because this paper is not concerned with the results of the Super Over. There were no matches of these types.

Each observation comprises a range of features (independent variables) and a response variable (dependent variable). In this case, the models are trying to capture whether the score at the halfway stage is sufficient. The dependent variable will, therefore, be the defending result, which will be set at 0 if the other team batting second scored a higher score, 1 otherwise.

3.2 Variable Description

The variables are captured mainly from the perspective of the defending team. In total, there are 13 independent variables and one response variable. The variables are grouped into the following categories:

- **Player Strength:** These are the measures of how strong the teams are from a batting and bowling perspective.
- **Stadium Information:** This is all the information about the stadium used in the match that is known before the match.
- **Match Information:** All other information known about the match.
- **Response Variable:** The variable that is being predicted.

Table 3.2 contains a breakdown of the variables used, including a broad group they can be classified into, the variable name and a description of the variable. In the actual model, the spaces in the variable name are replaced with underscores. It is just presented without them here to make it easier to read.

Ground capacity was considered as a possible feature, but the COVID-19 pandemic caused severe limitations on attendance during matches. Regardless, actual attendance figures were not available for every match.

Pitches are not required to be precisely in the middle of the ground and, therefore, can vary in distance to the boundaries on either side of the pitch. Each ground also has several pitches side-by-side with different pitches used during different matches, meaning distances to the boundary can vary between matches on the same field. After every over, teams switch the side from which they bowl, but if one side is shorter, it does tend to make a difference to scoring as teams target the shorter side for boundaries. Since there was no match-by-match data for the boundary length on each side of the pitch, or which pitch was used, it was decided to simply combine this into one figure, viz., ground width. Lengthwise, pitches tend to be more consistent on either side, so it was decided to also just use a ground length variable. This would also make it more consistent with the ground width variable.

Most of the variables used are self-explanatory. The player strengths are not standard measures, so they will need further explanation.

Group	Variable Name	Description
Match Information	Defending Team	The name of the team that batted first.
	Chasing Team	The name of the team that batted second.
	Time	The time when the match took place. 'Day' for matches starting before 3 pm, else 'Day/Night'.
	Defending Toss	Whether the defending team won the toss or not. 'Yes' or 'No'.
	First Innings Score	The score set by the team batting first.
Player Strength	Bat 2 Strength	Measure of the strength of the team batting second.
	Bowl 2 Strength	Measure of the strength of the team bowling second.
Stadium Information	Defending Stadium	Whether the defending team is 'Home', 'Away' or 'Neutral'.
	Stadium Median First Inn Score	Median score by the teams batting first for that particular stadium.
	Stadium Median Second Inn Score	Median score by the teams batting second for that particular stadium.
	Ground Altitude	The altitude of the stadium as measured in metres.
	Ground Length	The length of the playing area in the ground in metres.
	Ground Width	The length of the playing area in the ground in metres.
Response Variable	Defending Result	Whether the defending team did not lose in regular time. 0 if the defending team lost without any Super Over. 1 otherwise.

Table 3.2: Summary of variables modelled

3.2.1 Player Strength

A measure of the player strengths for the IPL needed to be developed. Given the success of Viswanadha et al. (2017) in their modelling of IPL matches using player strengths, their strength measures were largely incorporated with a few tweaks to make it more suitable for the task at hand here.

One big issue for player strength calculations is players who are new to the IPL. In the IPL, it is common to have many new signings every season. These players do not have any of the required player statistics. Viswanadha et al. (2017) did not mention their methodology for these cases. For these players, some sort of average needed to be assigned to them. A *phasing-in* approach could then be taken where as the players played more matches, more of their actual performance is used than the average to assign them a particular score.

Notation Used

For player strengths, both a *career score (c)* and a *form score (f)* were incorporated. The form score is defined in the same way as the career score but measured using the last n_f matches that a player has played in. The final score will then be a combination of the two. For the notation listed in Table 3.3 and used in the formulae of the player strength measures where there is a *(c)*, the *(f)* measure is defined similarly but using the last n_f matches that a player has played in.

Notation	Description
$MP_{x(c)}$	# Matches Played by Player x
$MBa_{x(c)}$	# Matches Player x has Batted In
$NO_{x(c)}$	# Innings Player x was Not Out
$RS_{x(c)}$	# Runs Scored by Player x
$BaB_{x(c)}$	# Balls Batted by Player x
$BaA_{x(c)}$	Batting Average of Player x
$BaSR_{x(c)}$	Batting Strike Rate of Player x
$BaSt_{x(c)}$	Batting Strength of Player x
$BaP_{x(c)}$	Batting Proportion of Player x
$WBaSt_{x(c)}$	Weighted Batting Strength of Player x
$WBaSt_x$	Net Weighted Batting Strength of Player x
n_f	# Matches Used in the Form Calculations
μ	Proportion of the Career Score used in the Final Calculation
$RC_{x(c)}$	# Runs Conceded by Player x
$BBo_{x(c)}$	# Balls Bowled by Player x
$OBo_{x(c)}$	# Overs Bowled by Player x
$OS_{x(c)}$	# Overs Started by Player x
$TOS_{x(c)}$	# Overs Started by Player x's Teams
$BC_{x(c)}$	# Bowling Contribution by Player x
$W_{x(c)}$	# Wickets by Player x
$BoE_{x(c)}$	Bowling Economy of Player x
$BoSR_{x(c)}$	Bowling Strike Rate of Player x
$BoP_{x(c)}$	Bowling Proportion of Player x
$WBoSt_{x(c)}$	Weighted Bowling Strength of Player x
$WBoSt_x$	Net Weighted Bowling Strength of Player x

Table 3.3: Notation used

Batting Strength

Batting average measures the average amount of runs contributed by a player for each dismissal of that player. It is defined as follows:

$$BaA_{x(c)} = \frac{RS_{x(c)}}{MBa_{x(c)} - NO_{x(c)}}$$

For the strength calculations, if a player has not batted in any of their innings, this is set to 0. If a player is not out in all their innings, they are given one out so that the denominator is 1.

Batting strike rate measures how quickly a player scores their runs. It is the number of runs they score per 100 balls faced.

$$BaSR_{x(c)} = \frac{RS_{x(c)}}{BaB_{x(c)}} \times 100$$

Similarly, this is set to 0 if a player has not batted in any of their matches.

Batting strength is the product of the batting average and batting strike rates. For one-day cricket, traditionally, the sum of these two measures is used, but Kumar (2014) argued that the product better captures the relative importance of each measure.

$$BaSt_{x(c)} = BaA_{x(c)} \times BaSR_{x(c)}$$

On average, not all batters are likely to contribute to a team's total; e.g. the number one batter certainly has more of an influence on a team's total than the number 11 batter, even if they were the same strength. The batting strength should, therefore, be weighted down by some measure of how often a player actually bats.

Batting Proportion is the square root of the proportion of matches batted in by player x.

$$BaP_{x(c)} = \sqrt{\frac{MBa_{x(c)}}{MP_{x(c)}}}$$

Weighted Batting Strength is the batting proportion of player x multiplied by the batting strength.

$$WBaSt_{x(c)} = BaP_{x(c)} \times BaSt_{x(c)}$$

Net Weighted Batting Strength (or simply *batting score*) is then a combination of the weighted career batting strength and weighted form batting strength.

$$WBaSt_x = \mu \times WBaSt_{x(c)} + (1 - \mu) \times WBaSt_{x(f)}$$

Bowling Strength

Bowling Economy is the average number of runs a bowler concedes per over

$$BoE_{x(c)} = \frac{RC_{x(c)}}{OBo_{x(c)}}$$

For the strength calculations, if a player has not bowled in any of their innings, this is set to 0.

Bowling Strike Rate is the average number of balls bowled by player x per wicket taken. This is set to 0 if no wickets have been taken by the player.

$$BoSR_{x(c)} = \frac{BBo_{x(c)}}{W_{x(c)}}$$

Bowling Strength is the inverse of the product of the bowling economy and bowling strike rate multiplied by some large constant. Without the constant, the result will be very small. The choice of constant is arbitrary and serves just to make the numbers more readable. A value of 100,000 is used here.

$$BoSt_{x(c)} = \frac{100000}{BoE_{x(c)} \times BoSR_{x(c)}}$$

If the denominator here is 0, then this is set to 0.

As with the batters, a weighted average is created. For the bowlers, this also has to account for how many overs they are expected to bowl. To do this, the new concept of *overs started* is introduced. For example, if a player bowls 3.3 overs and their team bowls 18.3 overs, it is said that the player has started 4 overs and the team has started 19 overs. In limited-overs cricket, the maximum number of overs each bowler is allowed to bowl is equal to the total number of overs divided by 5. So, in a 20-over match, each bowler is only allowed to bowl 4 overs each. For each match, a bowling contribution score is defined as follows:

$$BC_{x(c)} = min(1, \frac{OS_{x(c)}}{TOS_{x(c)}} \times 5)$$

Bowling Proportion is the square root of the ratio of bowling contribution to total matches played by player x.

$$BoP_{x(c)} = \sqrt{\frac{BC_{x(c)}}{MP_{x(c)}}}$$

Weighted Bowling Strengthis the bowling proportion of player x multiplied by the bowling strength.

$$WBoSt_{x(c)} = BoP_{x(c)} \times BoSt_{x(c)}$$

Net Weighted Bowling Strength (or simply *bowling score*) is then a combination of the weighted career bowling strength and weighted form bowling strength.

$$WBoSt_x = \mu \times WBoSt_{x(c)} + (1 - \mu) \times WBoSt_{x(f)}$$

Parameters Used

The parameters μ and n are set at 0.8 and 5, respectively. The former is the weight attached to the career scores over form scores. The latter is how many matches to use in the form score. Viswanadha et al. (2017) used values of 0.8 and 4, respectively. In their paper, these values were chosen using a grid search, and the one that yielded rankings closest to their estimated rankings in terms of least squared error was chosen. After some investigation to see how these parameters affected the number of matches played by the top-ranking batters under this system, it was decided to change the n parameter to 5. In future work, these parameters can possibly be tweaked.

New Players

Players' scores can vary greatly in the first few matches they play. To get around this, a phased approach is used where initially, players use an average score, and as the matches go on, their own scores get weighted more. The average here would have to be something specific to a player's expected output, e.g. a batter is more likely to contribute more to the batting total than a bowler, and therefore, assigning them an average specific to batters would make more sense. After five matches, players will have enough matches to have a full form score, so using five matches to scale in the players seems reasonable. The scaling for players will, therefore, be as per Table 3.4.

Matches Played	Own Score	Average Score
0	0%	100%
1	20%	80%
2	40%	60%
3	60%	40%
4	80%	20%
5+	100%	0%

Table 3.4: New player scaling

The averages would best be split by role. Depending on their strengths, players are split into the following four roles:

- **Batter:** These are players selected based on their batting strength and will very rarely, if ever, bowl.
- **Bowlers:** These are players selected based on their bowling strength. In cricket, all players may be required to bat in an innings if enough wickets have fallen, so although

these players also do bat often, the team is not largely dependent on their batting strength.

- **Batting Allrounders:** These are players who largely contribute in both the batting and bowling departments, with batting being their stronger quality.
- **Bowling Allrounders:** These are players who largely contribute in both the batting and bowling departments, with bowling being their stronger quality.

Furthermore, each player was classified as either a *local* (Indian) or *foreign* (Non-Indian) player. In the IPL, only four foreign players are allowed in a team at a time. Teams usually spend a lot of money to secure the best foreign players. As explained by Kampakis and Thomas (2015), new foreign players tend to perform better than new local players in their first few matches. The new foreign players tend to be among the best players in the world, while the new local players are usually young players making their first steps into professional cricket. This split will be referred to as the split by nationality. All the averages would, therefore, be split into the categories seen in Table 3.5.

	Batter	**Batting All-rounder**	**Bowling All-rounder**	**Bowler**
Indian	Local Batter	Local Batting All-rounder	Local Bowling All-rounder	Local Bowler
Non-Indian	Foreign Batter	Foreign Batting All-rounder	Foreign Bowling All-rounder	Foreign Bowler

Table 3.5: Player classifications into role and local/foreigner

AB de Villiers, the South African batter, for example, would be classified under Foreign Batters. Averages are calculated using the latest scores for every player in that category at the start of the match, e.g. If AB de Villiers played in matches 1, 4, 8, 18, 29, 36, and 45, then at the start of match 38, only his strength as at the end of match 36 would be used since that is the last match he played in before the current match. Only players who have played at least five matches were used in the calculation of category averages.

3.3 Data Preparation

3.3.1 Match Information and Player Strength

First, the scorecards for each match were extracted. This was done by looping through each link in the file with match links and web-scraping the information. Some matches were rained out after one innings, so exceptions were added to only link for one innings here.

When web-scraping the data, it was found that there was a period of matches where the batting orders were flipped. For these matches, exceptions were added to flip the batting order back to the original order. However, it was later found that for some of these, the order was not just flipped but also out of order in other ways, so a more manual reordering would need to be done. Besides that, it was also found that there were matches where there was no *did not bat* field in the scorecards. This field indicates the batting order of those players who did not have a chance to bat in the match. These two issues made it impossible to use any information about the actual batting order, so no methods that involved these could be used. For the bowling order, the same issue of the mixed orders was found. Bowling orders

also tend to vary a lot between matches. It, therefore, made more sense to use the player roles and countries for the averages instead of anything related to batting or bowling order.

Player roles and countries can be found online on various cricketing websites under each player's profile. All the matches were looped through again, but this time, going into each player's profile and getting their listed country and role. The Cricsheet website also contains an additional role of *wicket-keeper batter*. Since these models do not account for fielding strength differences, these players are simply classified as batters. A few of the players, mainly some older players and some players who did not play many matches, did not have their roles specified on the website. These were added in manually. Some of these players did have their roles listed on the ESPN Cricinfo website. ESPN Cricinfo does not differentiate between the two types of allrounders. For these allrounders, some research was needed to find their stronger suit. Similarly, additional research was necessary for those missing roles that were also not on ESPN Cricinfo.

Next, match scorecards were used to create two lists: a batting performance list and a bowling performance list. For each match, all relevant statistics from the match for each player were stored in the relevant list. Even if a player did not perform that specific action in the match, noting this was still necessary because of how often they had contributed to a team in that specific action was needed for the player strength calculations. Therefore, it had to be determined from the team lists which players did not bat or bowl in the match in order to fill this into the lists. Both these lists were then split by player so that each player had their own batting and bowling table. Several inconsistencies in the data first needed some special processing here:

- Inconsistent team names. Delhi Daredevils and Delhi Capitals were used interchangeably. Here, the team's actual name change started with the 2019 season. This was effectively still the same team, though, so it was decided to just change the name to Delhi Capitals throughout.
- Inconsistent names of players. Sometimes, in the same match, an extra name was used, or only initials were used for first names. These became apparent when looking at the actual batting and bowling information on the scorecard against the team list to see who did not bat or bowl in that match. Here, methods were added to identify names that matched and set the code to use the full name.
- Specific players with name issues. When adding the player roles to the player information the roles of certain players were not found. These were dealt with on a case-by-case basis. KL Rahul was referred to as Lokesh Rahul in some matches. Khaleel Ahmed was listed as K Khaleel Ahmed in some matches. Harmeet Singh Bansal was sometimes listed as Harmeet Singh, but another Harmeet Singh played in one match. Ravichandran Ashwin and Murugan Ashwin played for the same team, with the former referred to as Ashwin and the latter as M Ashwin in some matches. Shikar Dhawan and Rishi Dhawan also played in the same team, with the former sometimes referred to as Dhawan and the latter as Rishi Dhawan in some matches.

Whilst looping through the matches to create the batting and bowling performance lists, an additional match list was also created where each line had the match information, and the players who played so the player strengths in each match could be easily calculated. For each match, players have both a batting and bowling strength measure.

First, all the batting lists and then all the bowling lists are looped over, calculating the player strengths as at the end of each match. Player performances were also stored in their specific role and nationality file, e.g. AB de Villiers' performances were also stored in the Foreign Batters files. Those players without enough matches at any specific point in time were excluded from the average calculations. After this, for each match, the average player

strength was calculated at the start of the match by taking the strength of each player in that category in their latest match before the current match. This was then used to assign player strengths to those without enough information.

The strength at the start of the match has to be used for predictions. Player and average strength for that player's role and nationality were merged using the sliding scale, where a player's strength measure is solely based on their own performances after five matches. Each player now had a strength score heading into each match. This was then used in the match information with team lists to calculate the relative team strengths.

3.3.2 Stadium Information

Due to the limited number of matches played in many of the stadiums, the actual names of the stadiums were not amongst the variables used in the modelling. First, the median innings scores for each innings for each stadium are derived. Innings that were affected by DLS are excluded. This means matches where the first innings was completed fully but the second innings was reduced, were included in the calculation of the first innings medians but excluded for the second innings medians. These were looked at manually on a match-by-match basis for those matches where DLS was applied.

For each innings, a median after every match was calculated. The stadium medians were phased in over five matches like the player strengths. This means that if a stadium had less than five matches, a weighted average was taken with the overall median as at that point. It would have been ideal to average over similar stadiums only based on some criteria, perhaps only over stadiums in that country. However, given the limited size of the dataset, it was decided that an overall average would be best.

The dataset was split by stadium, and a median was calculated as at the start of each match based on previous matches in the stadium. The last completed match before every match was also calculated in order to find the overall median that would be used. These were then combined using the phasing-in approach above to calculate the appropriate medians.

Next, the dimensions and altitude of the stadium were considered. This information was not available online for most of the stadiums. For these measurements, the altitude and measurements were taken from *Google Earth* manually. For the stadiums where the information was available online, this was quite close to the measurements from Google Earth. The stadium width was not split because the pitch sometimes moves around through the pitch area. To be consistent, the pitch length was also taken as a whole. In general, the length of the pitch was also quite consistent on either side.

Finally, the home-ground advantage information was added, where it was determined if either team was playing at home or whether it was a neutral ground. For the following seasons, there was no traditional home-ground advantage:

- The 2009 IPL was held in South Africa due to the general elections in India over the same period. It was felt that security in India would not be able to cope with both events at the same time.
- The first half of the 2014 IPL season was held in the United Arab Emirates (UAE) for similar reasons.
- The 2020 IPL season was held in the UAE due to the COVID-19 pandemic.
- The 2021 IPL season started in India at limited venues but was moved to UAE roughly halfway through due to fresh scares of the COVID-19 pandemic.

- The 2022 IPL season was held in India but again at limited venues again due to the COVID-19 pandemic.

For all matches played in other countries, there were limited venues, and teams tended to move between venues. There was also no added advantage of home support in any of the venues for any of the teams in particular. All IPL matches played outside of India were classified as being played on neutral grounds for all teams involved.

Where matches were played in India, but the number of stadiums was limited, teams rotated amongst venues equally. Fan attendance was also drastically capped during most of that time because of the coronavirus pandemic. It could be argued that as the caps on attendance started easing, teams from the same city where a match would be played would enjoy more support. However, various other factors besides increased support could lead to home-ground advantage, as seen in the work by Gómez-Ruano, Pollard, and Lago-Peñas (2021) discussed in Section 2.3. Pune and Mumbai (the two cities used for venues) are also roughly just 152km away, and all teams trained in Mumbai (except Chennai Super Kings, who chose to train in Surat, approximately 278km outside Mumbai, in order to not have to share training grounds with other teams). This meant familiarity with the area and travel times were not an issue. All teams also played plenty of matches on each field as the tournament progressed, meaning familiarity with the actual conditions of each stadium and pitch for all teams involved. For these reasons, all IPL matches played in India, where the number of venues was limited, were classified as being on neutral grounds for all teams involved.

The assigned home team is always specified as the first team in the fixture. In order to determine which stadiums were the home stadium for which teams, all the round-robin matches in India were used as a start. The assigned home teams in the round-robins were then used to create a grid for all the teams and all the grounds, which measured how many times a team was assigned as the home team for each ground. If a ground had only one team with a positive number here, it would mean that that ground is exclusively used by that team as a home ground. For some grounds, there may be more than one team with a positive value. That means that this ground was shared. For most of these, the ground was used by the different teams in different years with no overlaps. That means that in a few seasons, the ground was used by one team and then taken over by another team. If a team no longer used a ground as their home ground, they would not be determined to have a home-ground advantage when playing there in the seasons that followed. In all the cases mentioned so far, the assigned home team will still be the actual team with the home-ground advantage. Lastly, there was the case where a ground was used by multiple teams in the same season. These cases were checked manually, and in none of these cases did two of these teams play each other at a shared ground. This means that there was no case where both teams would have been classified as being at home.

The playoffs and knockouts would need to be dealt with next. These matches are usually played at a predetermined ground. This means that grounds could be neutral or at a ground that one of the teams uses as a home ground. However, the team specified first is now whichever team finished in a certain position or won a particular match. Therefore, a team playing on their usual home ground may actually be specified second in the fixture. For each of the playoffs and knockouts, it was checked whether either of the teams had used that ground as a home ground during that season. Again, it was found that no two teams who both used a ground as a home ground during that season played each other at that ground. All fixtures now had a team with home-ground advantage, or the ground for the match was determined to be neutral. This was then converted for the final data and specified in terms of whether the defending team had a home-ground advantage or if the ground was neutral.

3.4 Exploratory Data Analysis

Exploratory data analysis (EDA) is an important step in any modelling exercise. Seltman (2012) classified EDA as, loosely speaking, any method of looking at data that does not include formal statistical modelling and inference. According to Natrella et al. (2010), EDA is used for the following reasons:

- Maximize insight into a dataset
- Uncover underlying structure
- Extract important variables
- Detect outliers and anomalies
- Test underlying assumptions
- Develop parsimonious models
- Determine optimal factor settings

For the EDA here, only the training data will be looked at to ensure that the test data can be used as truly unseen data and that it does not influence the modelling.

3.4.1 Categorical Variables

First, a breakdown of all the categorical variables for each team to verify that all teams are represented relatively similarly across all categories available at the start of the match.

Team	**Played**	**Defend/Chase**		**Home Ground Adv.**			**Time of Day**		**Toss**		**Result**	
		Defend	Chase	Home	Away	Neutral	Day	D/N	Won	Lost	Won	Lost
Chennai Super Kings (CSK)[1]	162	82	80	60	55	47	39	123	90	72	98	64
Deccan Chargers (HDC)[2]	45	27	18	24	21	0	12	33	24	21	17	28
Delhi Capitals (DC)[1]	175	78	97	67	67	41	45	130	91	84	81	94
Gujarat Lions (GL)	30	14	16	14	15	1	7	23	15	15	14	16
Kochi Tuskers Kerala (KTK)	13	7	6	6	7	0	3	10	7	6	6	7
Kolkata Knight Riders (KKR) [1]	177	81	96	67	70	40	43	134	91	86	96	81
Mumbai Indians (MI) [1]	190	104	86	73	70	47	43	147	100	90	116	74
Pune Warriors (PW)	45	20	25	23	22	0	18	27	20	25	13	32
Punjab Kings (PBKS) [1]	173	93	80	70	68	35	49	124	74	99	73	100
Rajasthan Royals (RR) [1]	142	66	76	54	52	36	41	101	77	65	67	75
Rising Pune Supergiant (RPS)	27	14	13	13	13	1	8	19	10	17	12	15
Royal Challengers Bangalore (RCB) [1]	174	87	87	64	71	39	48	126	82	92	85	89
Sunrisers Hyderabad (SRH) [1]	133	70	63	44	48	41	32	101	62	71	65	68

[1] Still active in the latest 2022 season.
[2] The full name of Deccan Chargers is Hyderabad Deccan Chargers, but this name is rarely used.

Table 3.6: Breakdown of categorical variables for each IPL team

Table 3.6 contains a breakdown of all the categorical variables per team. Gujarat Titans (GT) and Lucknow Super Giants (LSG) are excluded since they only debuted in the 2022 season. From Table 3.6 the following is noted:

- Almost all the current teams have played a large number of matches, which would make team name a valid variable.
- There is a fairly even spread amongst the current teams in terms of how many times they have defended and how many times they have chased.
- As expected, teams have played a fairly even number of home and away matches, given that each team plays each other home and away except for the playoffs. The large number of neutral matches are from those seasons played outside India.

- Day/night matches are played more regularly than day matches, with 73.89% of matches being day/night matches.
- Teams have a fairly even amount of luck with the toss. However, Chennai Super Kings seem to have won far more than their fair share of tosses (90 wins vs. 72 losses) and Punjab Kings seem to be particularly unlucky regarding tosses (74 wins vs. 99 losses). It will be interesting to see how important the toss is in the models.
- Some teams have clearly been more successful than others. It will be interesting to see if the actual teams are still used in the final models or if the player strengths can capture their differences.

Table 3.7 contains a breakdown of the response variable against the other categorical variables, excluding the team names. From this, the following is noted.

- Chasing teams seem to have an advantage, having won 56.41% of matches.
- Home teams seem to have an advantage, having won 54.77% of matches that were not played on neutral grounds.
- Winning the toss seems to only give a slight advantage (if anything), leading to wins in 52.9% of matches.

	Played	**Defend/Chase**		**Home Ground Adv.**			**Toss**	
		Defend	Chase	Home	Away	Neutral	Won	Lost
Winner	743	325	418	317	262	164	393	350

Table 3.7: Breakdown of winners for select categorical variables

3.4.2 Numeric Variables

First, a look at the correlation matrix, which shows the strength of the linear relationship between all the numeric variables in the data. Figure 3.1 contains the correlations between the numeric variables. There are no particularly strong correlations except perhaps between the ground length and ground width and, to a lesser extent, between the first innings median and second innings median. The former implies that grounds vary in size, but the shape remains relatively the same. The latter shows that high-scoring grounds for the team batting first also tend to be high-scoring for the team batting second.

Player Strength

Player strengths are calculated at the start of each match. In line with the purposes of EDA, it makes sense to just look at the player strengths that were actually used in the modelling. EDA on player strengths will only include player strengths during the training dataset and will not include the strengths that players have after they have finished their final match in the training dataset period. Player strengths are also not considered during the burn-in period to be consistent with EDA on the team strengths. During the burn-in period, the team strengths will not be useful because many players would not have played many matches, and the overall strengths per category will also be quite volatile. Player scores are also quite volatile during their first few matches, so only those who have played at least 10 matches are included here.

Table 3.8 looks at the top 10 net weighted batting strengths heading into a match. The batting number listed there is the number in their next match.

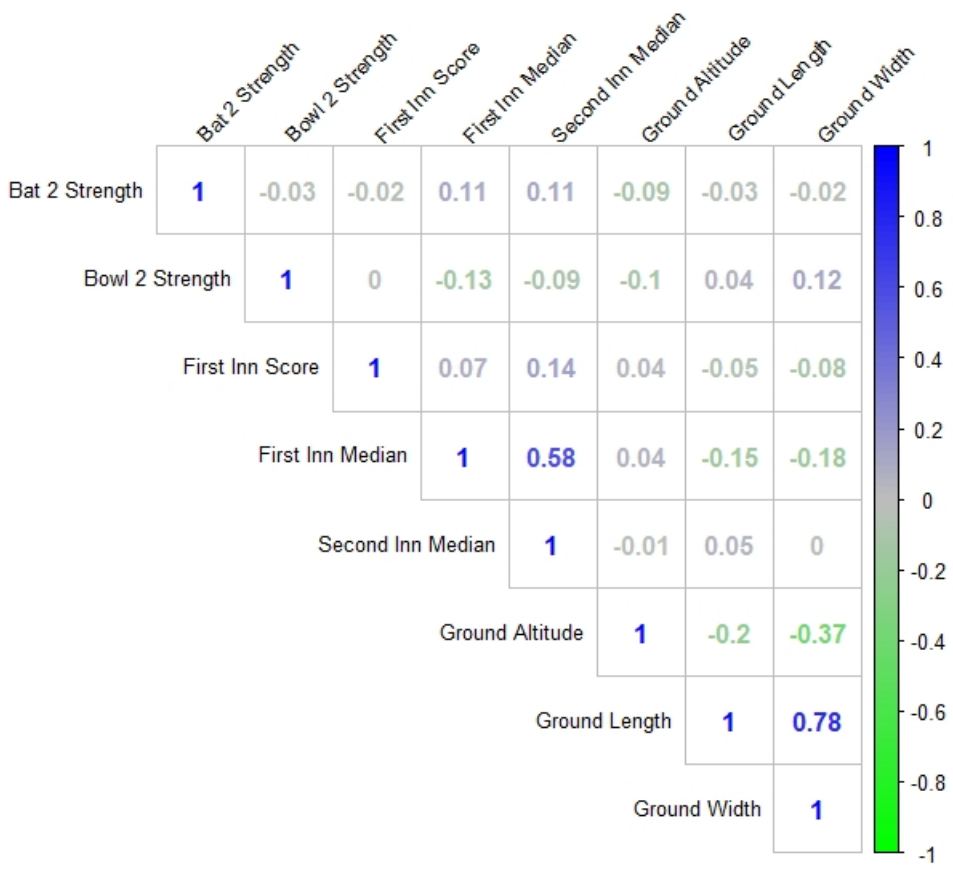


Figure 3.1: Numeric feature correlations

Num	Player	Country	Match Date	Team	Role	BatNum	Matches	Batting Score
1	David Miller	South Africa	2013-05-11	PBKS	Batter	5	13	14 091.82
2	Chris Gayle	West Indies	2013-04-29	RCB	Batter	2	50	13 241.36
3	MS Dhoni	India	2018-05-11	CSK	Batter	4	167	12 321.56
4	MS Dhoni	India	2018-05-13	CSK	Batter	4	168	12 060.22
5	Vijay Shankar	India	2019-04-04	SRH	Batting Allrounder	3	20	11 289.34
6	David Miller	South Africa	2013-05-14	PBKS	Batter	4	14	11 205.48
7	David Miller	South Africa	2014-04-26	PBKS	Batter	5	20	10 782.87
8	Chris Gayle	West Indies	2013-05-06	RCB	Batter	2	52	10 689.33
9	Jacques Kallis	South Africa	2010-03-31	RCB	Batting Allrounder	2	31	10 643.83
10	Chris Gayle	West Indies	2013-04-09	RCB	Batter	2	43	10 601.23

Table 3.8: Highest batters strengths (minimum 10 matches)

To put the scores into context, Figure 3.2 looks at a boxplot of all the individual batting scores for players who have played at least 10 matches split into the different roles.

Table 3.9 shows the strongest teams in terms of batting strength. Again, this will be for values that were actually used in the modelling, so it will only be for teams that batted second in a match that met the requirements to be part of the training data.

Figure 3.3 presents a histogram of all chasing team batting strengths going into matches in the training set. A *five-number summary* of this data is given as 17,220.58 (minimum), 24,653.79 (first quartile), 26,975.13 (median), 29,659.68 (third quartile), 40,867.71 (maximum). This helps contextualise how strong the strongest scores presented in Table 3.9 are. The data may appear to resemble a normal distribution visually. However, a *Shapiro–Wilk Test* (which is a test of normality) on the data gives a p-value of 0.0006, which is enough to reject the null hypothesis that the data comes from a normal distribution with a great degree of confidence.

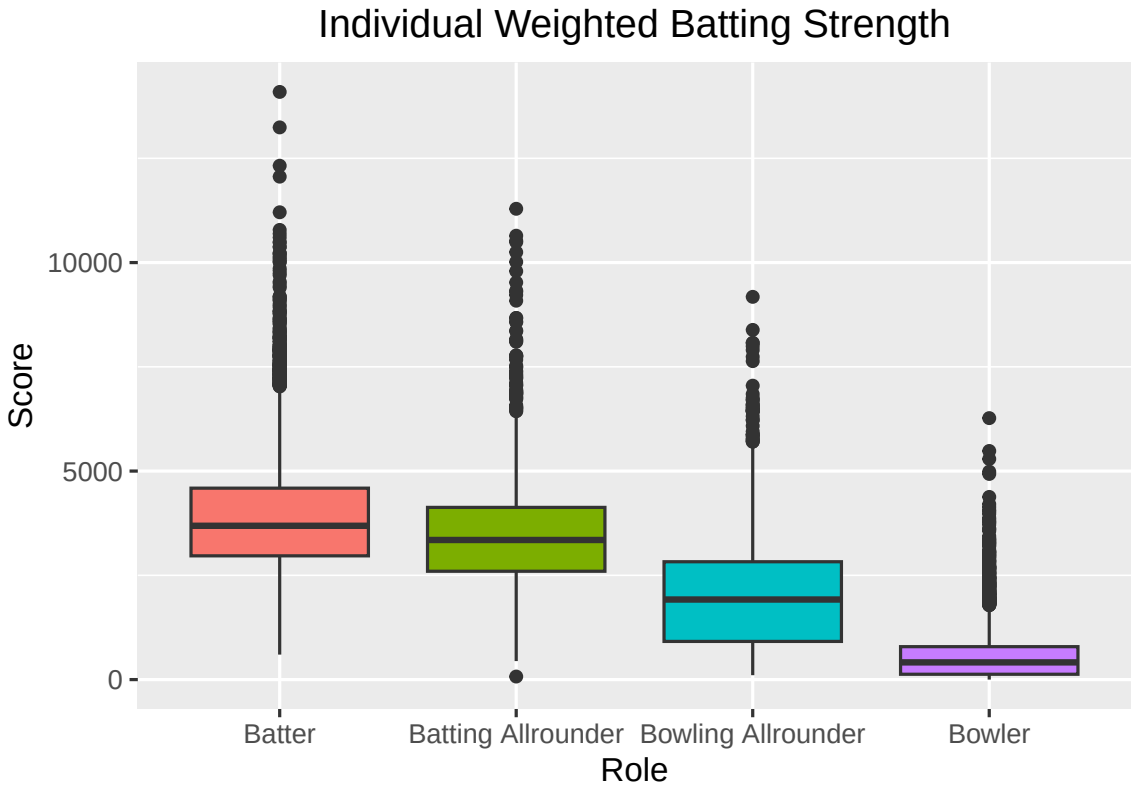


Figure 3.2: Boxplot of weighted batting strengths (minimum 10 matches)

Num	Team	Opponent	Date	Bat 2 Strength
1	Mumbai Indians	Kolkata Knight Riders	2016-04-28	40 867.71
2	Royal Challengers Bangalore	Sunrisers Hyderabad	2016-05-29	38 445.21
3	Chennai Super Kings	Royal Challengers Bangalore	2021-09-24	38 405.87
4	Sunrisers Hyderabad	Delhi Capitals	2019-04-14	38 316.90
5	Chennai Super Kings	Royal Challengers Bangalore	2018-05-05	37 802.35
6	Chennai Super Kings	Royal Challengers Bangalore	2011-05-24	37 467.12
7	Royal Challengers Bangalore	Delhi Capitals	2016-05-22	37 398.70
8	Sunrisers Hyderabad	Delhi Capitals	2019-04-14	37 310.3
9	Royal Challengers Bangalore	Gujarat Lions	2016-05-24	37 260.30
10	Royal Challengers Bangalore	Punjab Kings	2015-05-13	37 139.22

Table 3.9: Team batting strength for chasing teams

Next, a similar breakdown for the bowlers. Table 3.10 looks at the top 10 bowling strengths heading into a match for players who have played at least 10 matches. The bowling number listed there is the number in their next match.

Table 3.10 also interestingly shows a batter, Sachin Baby, appearing twice. Baby bowled for the first time in his tenth match and only bowled four balls but took two wickets for four runs. This gave Baby a really low bowling strike rate, which meant a really high bowling strength. Even with the factor that weighted the bowling strength by how much of the team's bowling that the player actually does, it was still enough to put Baby on top of the bowling rankings. The only other match that Baby bowled was his 12^{th} match when he bowled one over for four runs with no wickets. Baby is clearly not a bowler, yet his score greatly increased his team's strengths, especially in his 11^{th} and 12^{th} matches. However, it was only in the 11^{th} match that this would have had an impact, as Baby's team batted second in his twelfth match.

One way to address this was to enforce a minimum number of balls that a bowler should bowl before they got a full score of their own. The issue is that this would cause inconsistencies

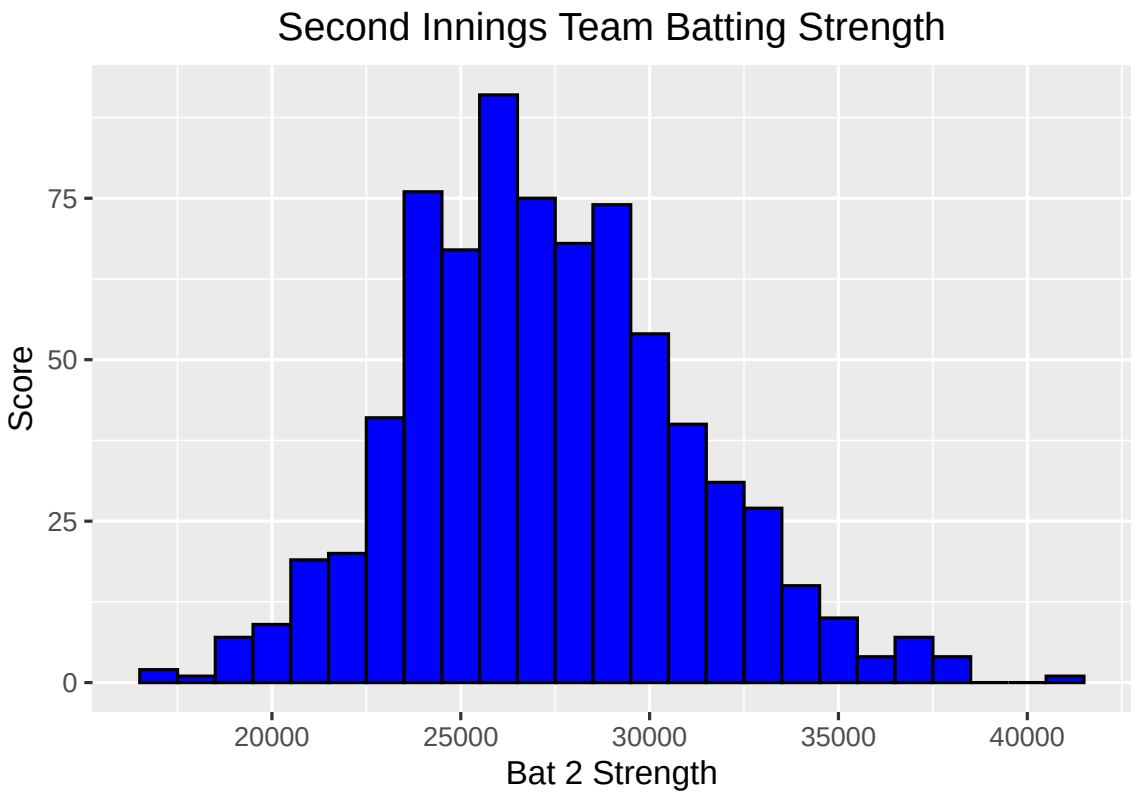


Figure 3.3: Histogram of chasing team weighted batting strengths

Num	Player	Country	Match Date	Team	Role	BowlNum	Matches	Bowling Score
1	Sachin Baby	India	2016-05-18	RCB	Batter	8	10	1 463.84
2	Sunil Narine	West Indies	2013-04-11	KKR	Bowler	3	16	1 460.38
3	Sunil Narine	West Indies	2012-05-22	KKR	Bowler	4	12	1 451.61
4	Sunil Narine	West Indies	2012-05-27	KKR	Bowler	3	13	1 431.13
5	Sunil Narine	West Indies	2013-04-03	KKR	Bowler	4	14	1 427.48
6	Sachin Baby	India	2016-05-22	RCB	Batter	8[1]	11	1 413.51
7	James Faulkner	Australia	2013-05-05	RR	Bowling Allrounder	2	10	1 410.84
8	Sunil Narine	West Indies	2013-04-14	KKR	Bowler	3	17	1 352.30
9	Sunil Narine	West Indies	2014-04-24	KKR	Bowler	3	32	1 326.92
10	Sandeep Sharma	India	2014-05-14	PBKS	Bowler	1	10	1 323.45

[1] Did not bowl in this match - shown is the number bowled at in the previous match.

Table 3.10: Highest bowler strengths (minimum 10 matches)

with how the batting score was calculated, and it would cause complications with the current scaling-in method that uses the number of matches played. In the end, since it was only one match where this score had an impact on training, it was decided to refrain from any manual intervention. It is unlikely that such a situation would arise again.

To put the scores in Table 3.10 into context, Figure 3.4 looks at a boxplot of all the individual bowling scores for players who have played at least 10 matches split into the different roles.

Table 3.11 shows the strongest teams in terms of bowling strength. Again, this will be for values that were actually used in the modelling, so it will only be for teams that bowled second in a match that met the requirements to be part of the training data.

Figure 3.5 looks at a histogram of all defending team bowling strengths going into matches in the training set. A five-number summary of this data is given as 2,338.89 (minimum), 3,237.98 (first quartile), 3,609.97 (median), 3,932.54 (third quartile), 5,648.39 (maximum). This helps contextualise how strong the strongest scores presented in Table 3.11 are. Again, the data may appear to resemble a normal distribution visually, but a Shapiro–Wilk Test on

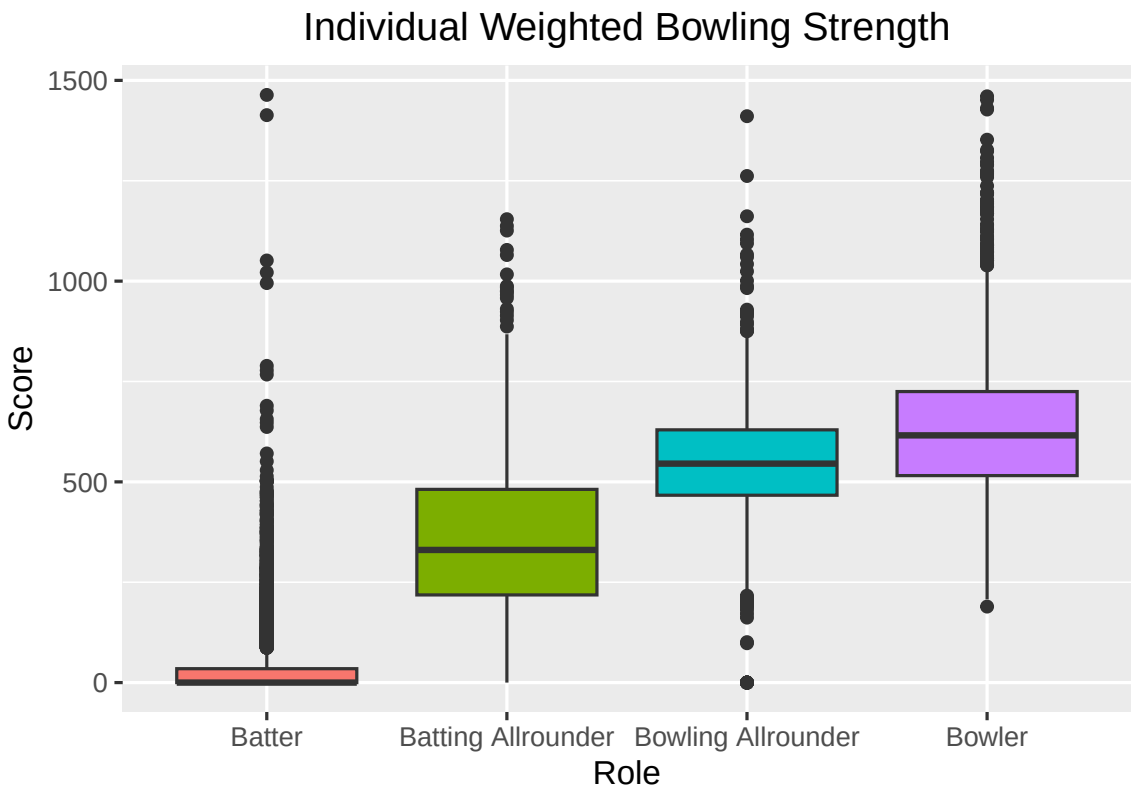


Figure 3.4: Boxplot of weighted bowling strengths (minimum 10 matches)

Num	Team	Opponent	Date	Bowl 2 Strength
1	Rajasthan Royals	Chennai Super Kings	2013-04-22	5 648.39
2	Kolkata Knight Riders	Delhi Capitals	2012-05-22	5 384.09
3	Rajasthan Royals	Royal Challengers Bangalore	2023-04-20	5 365.44
4	Mumbai Indians	Rajasthan Royals	2013-05-15	5 058.01
5	Rajasthan Royals	Mumbai Indians	2013-04-17	5 007.55
6	Mumbai Indians	Rajasthan Royals	2011-05-20	4 978.89
7	Kolkata Knight Riders	Pune Warriors	2012-05-19	4 919.76
8	Sunrisers Hyderabad	Rajasthan Royals	2013-04-27	4 861.73
9	Mumbai Indians	Royal Challengers Bangalore	2012-05-09	4 795.89
10	Kolkata Knight Riders	Delhi Capitals	2014-04-19	4 757.01

Table 3.11: Team bowling strength for defending teams

the data gives a p-value of 0.004, which is enough to reject the null hypothesis that the data comes from a normal distribution with a great degree of confidence.

First Innings Score

Figure 3.6 shows a breakdown of how the defending team did in terms of the result for different first innings scores. There is a large number of scores in the middle, where it seems the result can go either way, whereas in the tails, the results tend to be quite clear-cut. 49 of the 51 scores under 119 lead to losses. On the other end, 22 of the 24 scores over 218 lead to victories, with all 14 of those over 226 lead to victories.

Median Scores

Table 3.12 summarises the scores involved. Note that these are the median scores for both innings, which vary by stadium at each point in time. This table shows that scores can vary greatly between the various stadiums. Some of the reasons cricket scores vary so much among stadiums are differences in the field sizes, pitch conditions, weather and altitude.

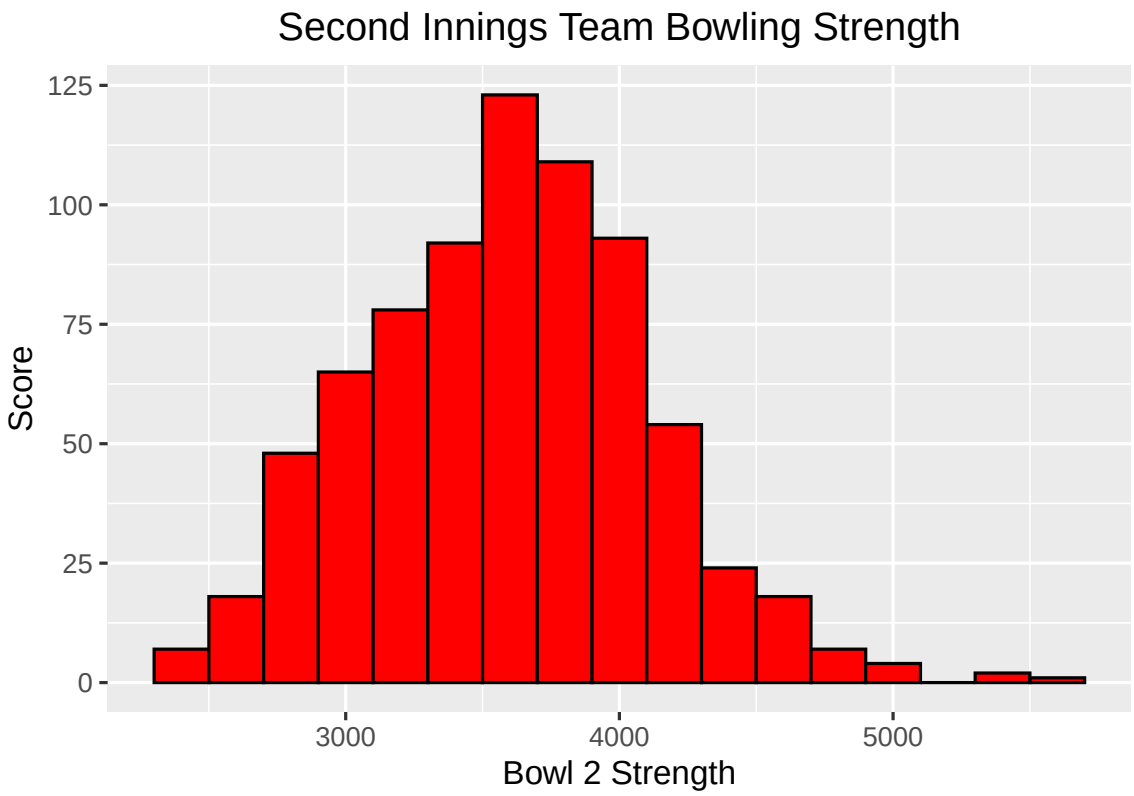


Figure 3.5: Histogram of defending team weighted bowling strengths

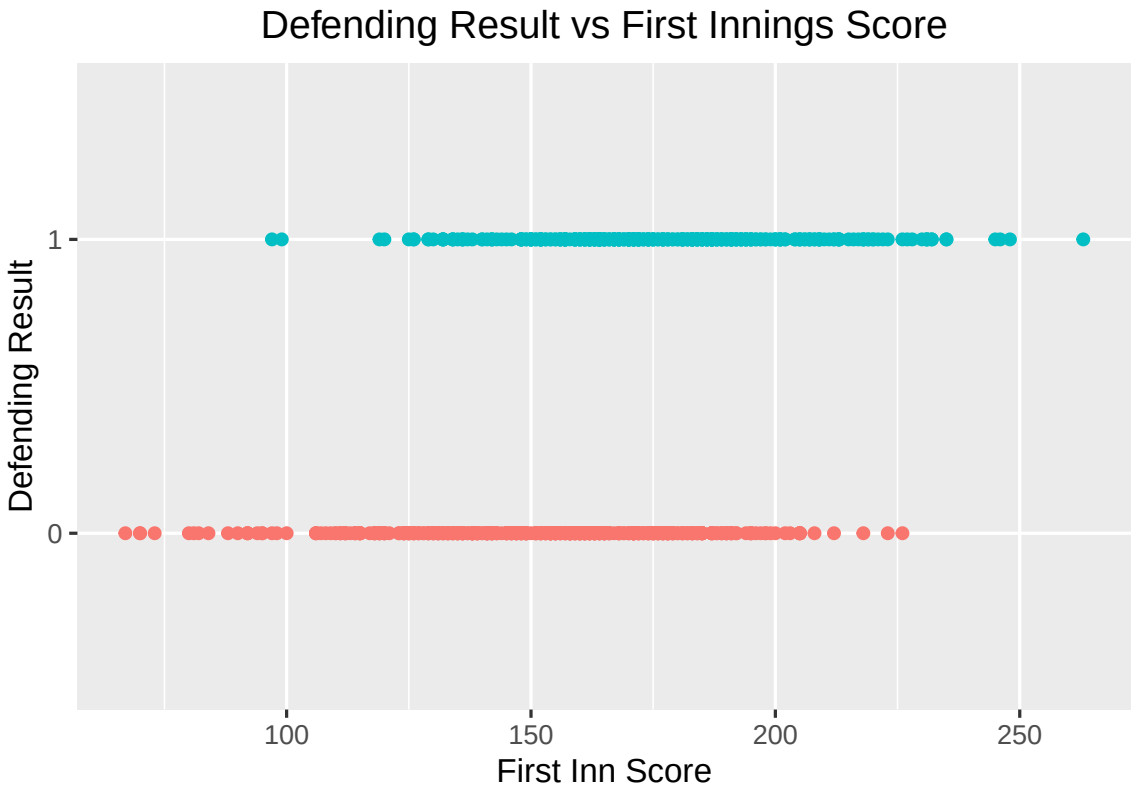


Figure 3.6: Defending result vs. first innings scores

Stadium Information

All information about stadiums except the median scores was entered manually based on the information added online. There are too many stadiums involved to break down all the

	First Innings Median	Second Innings Median
Minimum	134.5	115.5
Quartile 1	157.3	144
Median	162	148
Mean	161.8	148
Quartile 3	167	152.5
Max	192	180

Table 3.12: Median scores per innings across stadiums

stadium information for every stadium. Instead, a summary of the stadium information is provided in Table 3.13. Ground capacity and ground area were not actually used in the training set but instead were used to show how drastically the sizes varied between stadiums. Ground area is just an estimate assuming the ground is a perfect oval and using the formula:

$$Area = \pi \times \frac{length}{2} \times \frac{width}{2}$$

	Ground Capacity	Ground Area (m^2)	Ground Altitude (m)	Ground Length (m)	Ground Width (m)
Minimum	11 000	10 074	2.0	118.9	107.9
Quartile 1	25 000	13 848	13.5	135.0	127.2
Median	31 000	14 655	113.5	137.5	137.0
Mean	38 036	14 917	368.8	140.7	134.5
Quartile 3	49 500	16 677	537.8	146.0	140.0
Max	132 000	18 787	1 621.0	172.2	160.1

Table 3.13: IPL Stadium dimensions

3.5 Conclusion

This chapter detailed everything related to the data used in this project. It started by giving some background into the raw data and explained how matches were selected for the training and test datasets. Next, the actual variables used were presented, along with how they were derived from the data. Exploratory data analysis was also performed to gain a greater understanding of the data used.

A deeper understanding of the data was given in this chapter, which then enables proper modelling of the data. In the following chapter, Chapter 4, the methodology used is explained, and then in Chapter 5, the actual modelling and analysis of results follows.

Chapter 4

Methodology

In Chapter 3, a basis for this chapter was provided by explaining the data used and how the relevant features were extracted. This chapter describes the methodology used in this study. It begins by outlining the modelling approach, including detailing how the data is split, the type of models used, and the evaluation criteria chosen. In the next chapter, the actual results of the models are looked at.

4.1 Modelling Approach

The modelling approach can be broken down into three stages, i.e. data split, model application and evaluation criteria.

4.1.1 Data Split

The latest completed season of the IPL (2022) will be used as the test set. The training set will be from the third season of the IPL until the season before the test set (2010-2021). Some of the features used require a certain number of historical games, and some of them will tend to be quite volatile over a short number of games. The first two seasons, therefore, form part of the calculation of features but are not used in the training set. The first two seasons are referred to as the *burn-in* data. Depending on the model, the data may need first to be standardised.

Standardisation

Standardisation essentially puts all the variables on the same scale. This removes arbitrary choices in units of measurement used, e.g. should the field measurements be expressed in metres or kilometres? If kilometres were used, for example, then the coefficient that deals with this variable would need to be much larger. This will not work well with regularisation, where there is a penalty for the size of every coefficient.

To standardise a variable, x_i the following formula is used:

$$z = \frac{x_i - \mu_i}{\sigma_i}$$

In the above, μ_i and σ_i are derived from the training set, and then the same scaling is applied to the test set for the corresponding variable. No information from the testing set is used when scaling because this should remain unseen data.

4.1.2 Method Application

Suitable models have to be chosen. These models each come with their own hyperparameters and parameters that have to be chosen/optimised. A model's parameters are the values that are learned during training. These values are estimated from the training data. An example

of a model's parameters would be the weights of a neural network. Hyperparameters are values that are set before the training begins and control aspects of the training process. Often, these hyperparameters are chosen using a grid search - a range of values for each hyperparameter is chosen, and then all possible combinations between the values in the different hyperparameters are tested to see which combination performs the best. Again, this tuning is done only on the training data.

The models used along with their evaluation approach used during hyperparameter tuning can be seen in Table 4.1. *Cross-validation (CV)* and *out-of-bag (OOB)* are terms seen in the table that are explained in the next section. As can be seen in the table, it is only the tree-based models that make use of out-of-bag. Naive Bayes will not use either because there are no hyperparameters that will be set from the training data in this paper.

Model	**Standardisation**	**Evaluation Approach**
Logistic Regression	Yes	Cross-Validation
Classification Tree	Yes	Cross-Validation
Boosting	No	Out-Of-Bag
Random Forest	No	Out-Of-Bag
Bagging	No	Out-Of-Bag
Support Vector Machines	Yes	Cross-Validation
Neural Networks	Yes	Cross-Validation
Naive Bayes	Yes	n/a

Table 4.1: Models and their evaluation approaches used during hyperparameter tuning

4.1.3 Evaluation Criteria

There must be a consistent measure that helps pick these hyperparameters and parameters and also allows comparison of the different models. Models will be compared on *accuracy*: the percentage of correct predictions. This is simply the number of correct predictions divided by the total number of predictions. Three different types of accuracies will be referred to in this paper: test accuracy, CV accuracy and OOB accuracy. The test accuracy is simply the accuracy on the test data set. The other two are methods that allow one to train on different subsets of the same data. They are both used to estimate performance on unseen data, which is useful when selecting hyperparameters. Both of these methods are explained a bit more below. When referring to *expected values* it simply means the average value over a large number of observations.

Some models use the *error rate*, which is the percentage of incorrect predictions. The error rate is, therefore, one minus the accuracy. Maximising the accuracy is, therefore, the same as minimising the error rate.

In addition to the test accuracy, the Brier score (see Section 4.1.3) will be used when comparing the various models' final performance. Brier score is not used in the training of hyperparameters or parameters.

Cross-Validation

k-fold cross-validation, also referred to simply as cross-validation, is the first of these methods that allow one to train on different subsets of the same data.

Berrar (2018) explained that cross-validation is a data resampling method that is generally used to assess the ability of models to generalise and prevent overfitting. It belongs to the Monte Carlo family of simulations. Overfitting is where models are made to fit the training

data too closely. Essentially, a model is built that perfectly fits the training set but does not generalise very well on new unseen data. Generalisation is best measured on new data. However, when data is already sparse, this is not practical.

Without cross-validation, the training set would have to be further split into another training and testing set. The new training set would be used to train models, and the new test set would be used to choose hyperparameters. The original test set must be kept as unseen data. The models used are then also affected by how the training set is split, with some valuable information that could be used in modelling lost. With cross-validation, the training set is split into $k$folds, with each fold acting as a test set in a different run. The model is then run k times, and the accuracy in each case is recorded. The average accuracy over all runs is referred to as the CV accuracy. Models are run with different hyperparameters, and the one with the highest CV accuracy is chosen. Figure 4.1 shows this process. The choice of k is somewhat arbitrary. The choice of $k = 10$ is a popular one and is the value of k that will be used in this paper.

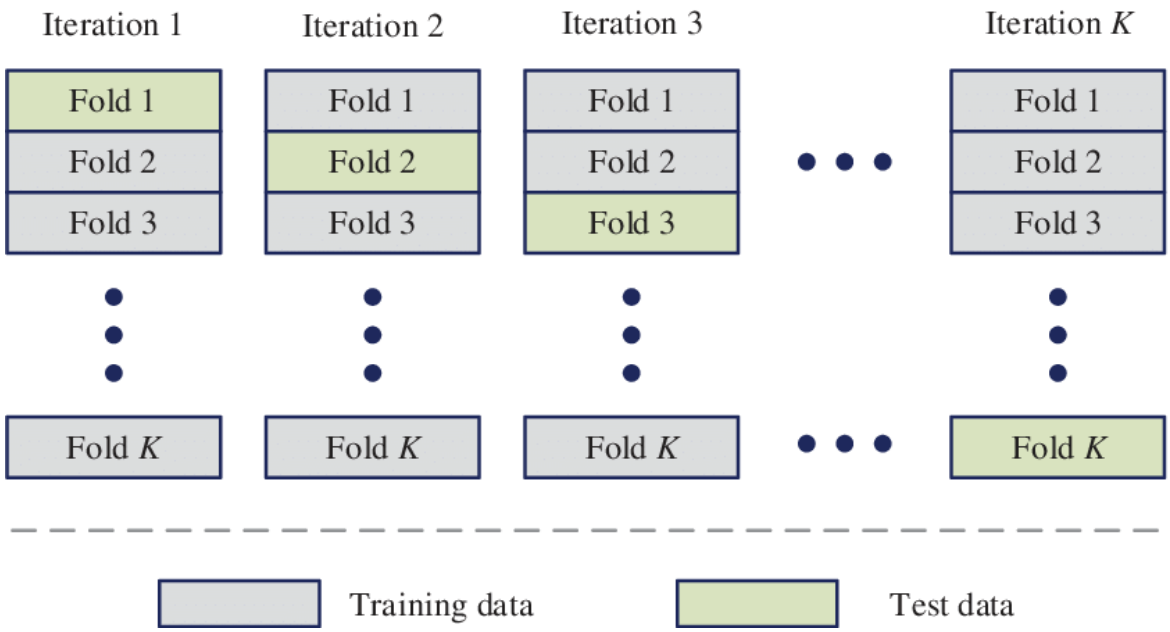


Figure 4.1: k-Fold cross-validation process

Source: Ren, Li, and Han (2019)

Out-of-Bag

Out-of-bag is the next method that is used to train on different subsets of the same data. Out-of-bag is used in tree models and involves picking a random subset of the data in every run and then measuring the accuracy on the observations not used. This concept is explained more in Section 4.4.1.

Brier Score

The Brier score measures the accuracy of probabilistic predictions based on the work of Brier et al. (1950). It measures the mean squared difference between the predicted probability and the actual outcome. The formula only works for binomial response variables where there are only two possible outcomes, as is the case here. Brier scores are values between zero and one, with 0 indicating perfect accuracy and 1 indicating complete inaccuracy. The formula for Brier score is as follows:

$$Brier\ score = \frac{1}{n} \times \sum_{t=1}^{n} (O_t - P_t)^2$$

where:

n is the number of samples

O_t is the actual outcome of the event for the t^{th} sample (1 if it happens, 0 if it does not).

P_t is the predicted probability of the event happening for the t^{th} sample.

A lower Brier score means that the model performs better. The difference between the accuracy and Brier score is that Brier score takes the probability of predictions into consideration. For example, consider the case where a coin that always lands on heads is flipped many times, and there are two models: model A says there is a 0.9 probability of a heads and model B says there is a 0.6 probability of a heads, then model A will clearly have a lower Brier score (it can be shown that the expected Brier score in model A is 0.01 and in model B is 0.16). If, however, it was a fair coin with a 0.5 probability of landing on heads, then there would be expected to be an equal number of heads and tails. and model B would be expected to have the lower Brier score (it can be shown model A has an expected Brier score of 0.41 and model B has an expected Brier score of 0.25).

A random prediction (where both outcomes have a probability of 0.5) is expected to yield a Brier score of 0.25. Consider the Brier score calculation in this case

$$Brier\ score = \frac{1}{n} \times \sum_{t=1}^{n} (O_t - 0.5)^2$$

The term $(O_t - 0.5)^2$ will always be 0.25 regardless if O_t is 0 or 1. The equation therefore becomes

$$\begin{aligned} Brier\ score &= \frac{1}{n} \times (0.25 \times n) \\ &= 0.25 \end{aligned}$$

A Brier score of 0.25 or above is therefore not considered to be a good predictor.

Brier scores are also generally sensitive to class imbalances in the dataset. For example, a model may be quite good at predictions, but if there is a dataset where almost all the observations come from category 1, a model that simply always predicts category 1 may have a better Brier score than the initial model. With this dataset, the classes are quite balanced, and this should, therefore, not be a problem.

Analyse of Evaluation Criteria

Consider a binomial variable that has x probability of being in category 1 and a model A that says that variable has x probability of being in category 1. The model therefore perfectly models the underlying distribution. It can be shown that if x was 0.5, model A will have an expected Brier score of 0.25 and expected accuracy of 50%. If x was 0.6, this changes to 0.24 for the expected Brier score and 60% for the expected accuracy. If x was 0.7, this changes to 0.21 for the expected Brier score and 70% for the expected accuracy. Probabilities of categories are symmetric and so values of *1 - x* in the above will yield the same Brier scores and accuracy as *x*. This illustrates that even if the model perfectly models the underlying distribution of a binomial variable, it will have a lower expected Brier score and expected accuracy the closer to 0.5 the underlying probabilities are.

*T*here are no absolute cut-offs for what a good value is for the Brier score - it will depend on the underlying data. Brier score is still a useful tool to compare different models on the same underlying data. Similarly for the accuracy. Results will also be presented visually in prediction probability ranges. This will help understand how the models performed in different ranges. For example, suppose a model predicts a 60% probability of avoiding a loss in regular time for the team batting first. In that case, only six out of 10 matches that fall in that category ending with a loss avoided in regular time for the team batting first is consistent with the model's predictions. This visual representation will not be used to choose the best performing model but rather just to understand the predictions better.

4.2 Logistic Regression

Logistic regression can be used to model binary dependent variables. It is similar to linear regression, except instead of a straight line, an "S"-shaped curve called a sigmoid is fitted like the one shown on the right-hand side of Figure 4.2. This diagram also shows why it is preferred to linear models for binary classification. Logistic regression allows the prediction of a probability that shows how likely an observation is to lie in a certain category.

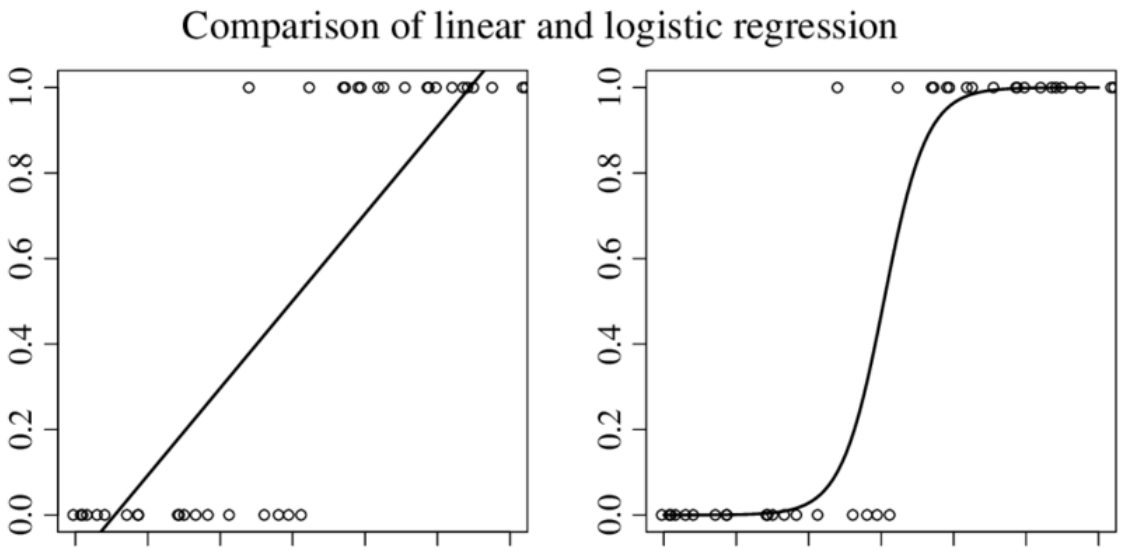


Figure 4.2: Linear vs. logistic regression for classification

Source: Adapted from Holliday (2012)

Define a model

$$P(Y = j \mid X = x) = \begin{cases} \pi(x) & \text{if } j = 1 \\ 1 - \pi(x) & \text{if } j = 0 \end{cases}$$

In the above, π(x) represents the probability that an observation is in category 1 given $X = x$.

The logistic map is defined as

$$\pi(x) = f(X) = \frac{e^t}{1 + e^t} \; where \; t = \beta_0 \; + \beta 1 X_1 + ... + \beta_p X_p$$

This logistic map is a simple sigmoidal function that is asymptotic to 0 and 1. A decision threshold is then set such that if π(x) $\geq$ the decision threshold, then 1 is deemed to be the most likely outcome, and that is set as the prediction. Otherwise, the prediction is 0. The default value for the decision threshold is 0.5, but it can also be tweaked.

4.2.1 Decision Threshold

There are specific cases where there may be a need to change the decision threshold from the 0.5 default value. Possible cases include where there are class imbalances in the dataset or where the costs of *false positives* and *false negatives* are different. False positives, in this case, are where the model says that the defending team will win, but they do not and false negatives are where the model says the defending team will lose, but they do not.

For this paper, the default threshold of 0.5 will be kept. In this dataset, the classes are balanced. Furthermore, the costs of false positives and false negatives are the same because the model can essentially be used by both defending and chasing teams. A false negative may not be bad for a defending team but will be for the chasing team. Furthermore, keeping the default threshold allows the output of the model to be interpreted as a probability. This enables the evaluation of model performance in different probability ranges that are used in Chapter 5.

4.2.2 Odds

The logistic function can be interpreted in terms of the "odds" of one event vs. another.

$$odds = \frac{\pi}{1-\pi}$$

This helps with the interpretation of logistic regression models (coefficients) because:

$$\begin{aligned} odds(X=x) &= \frac{\pi(X=x)}{1-\pi(X=x)} \\ &= \frac{Pr(Y=1|X=x)}{Pr(Y=0|X=x)} \\ &= e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p} \end{aligned}$$

This is referred to as the odds that $Y=1$ vs. $Y=0$ given that $X=x$ and essentially says how much more likely it is that $Y=1$ than $Y=0$ given that $X=x$.

Next consider what happens to the odds when X_1 increases by 1 unit and goes from $X_1=x_1$ to $X_1=x_1+1$ while all other predictors remain constant

$$\begin{aligned} odds(X_1=x_1+1) &= e^{\beta_0+\beta_1(x_1+1)+\ldots+\beta_p x_p} \\ &= e^{\beta_1} e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p} \\ &= e^{\beta_1} odds(X_1=x_1) \end{aligned}$$

Put in another way.

$$e^{\beta_1} = \frac{odds(X_1=x_1+1)}{odds(X_1=x_1)}$$

The left-hand side of this equation is known as the *odds ratio*. This gives a way to interpret the impact of an independent variable on the dependent variable in logistic regression. An increase in X_1 by 1 unit (whilst all other predictors remain constant) increases the odds of $Y=1$ by a multiple of e^{β_1}. The β values are therefore also known as the log-odds.

The question then becomes, what if X_1 was a categorical variable? Consider an example categorical variable car colour, which can either be red, blue or green. For categorical variables, a base category is chosen arbitrarily. E.g. suppose the base colour is red in this case. There are then two new variables created: car colour: blue and car colour: green, which both take on the value of 1 if a car is that colour. If the car is red, both of these variables will be 0. The coefficient of these variables then gives us their log-odds. If car colour: blue had a coefficient

of β_a for example, then the car being blue increases the odds of $Y = 1$ by a multiple of e^{β_a} than if the car was red. β_0 gives the log-odds when all categorical variables have their base value and all numeric values are 0.

4.2.3 Regularisation

Lasso regression will be used to prevent overfitting. Lasso regression, also known as L1 regularisation, is a regularisation technique. Regularisation techniques put penalties in proportion to the coefficients used when calculating the mean square error (MSE). MSE is the average of the squared differences between actual and predicted values. This forces the coefficients to be shrunk, which then reduces the variance of the model and prevents overfitting. Cross-validation is used to select the optimal penalty. The optimal penalty is defined as the one that leads to the lowest CV error. Sometimes, the penalty to be used is taken as one standard error from this point. This is known as the *one standard error rule* and has various justifications as to why it should rather be used. However, this paper will stick to the much more understandable minimum CV error point.

Lasso regression aims to minimise the value of

$$\sum_{i=1}^{N}(y_i - (\beta_0 + \sum_{j=1}^{p}\beta_j X_j))^2$$

subject to the constraint

$$\sum_{j=1}^{p}|\beta_j| \leq \tau$$

A common alternative to Lasso regression that will also be used in this paper is Ridge regression, also known as L2 regression. The value to be minimised is the same in Ridge regression, but the constraint then becomes

$$\sum_{j=1}^{p}\beta_j^2 \leq \tau$$

4.2.4 Advantages/ Disadvantages

For logistic regression, the advantages and disadvantages are the following:

Advantages

- Easy to implement and interpret.
- Makes no assumptions about the distribution of independent variables.
- Weights give an indication of the importance of variables as well as the direction of the relationship with the dependent variable.

Disadvantages

- Assumes a linear relationship between the dependent and independent variables.
- Performance affected if independent variables are correlated.
- Sensitive to outliers.

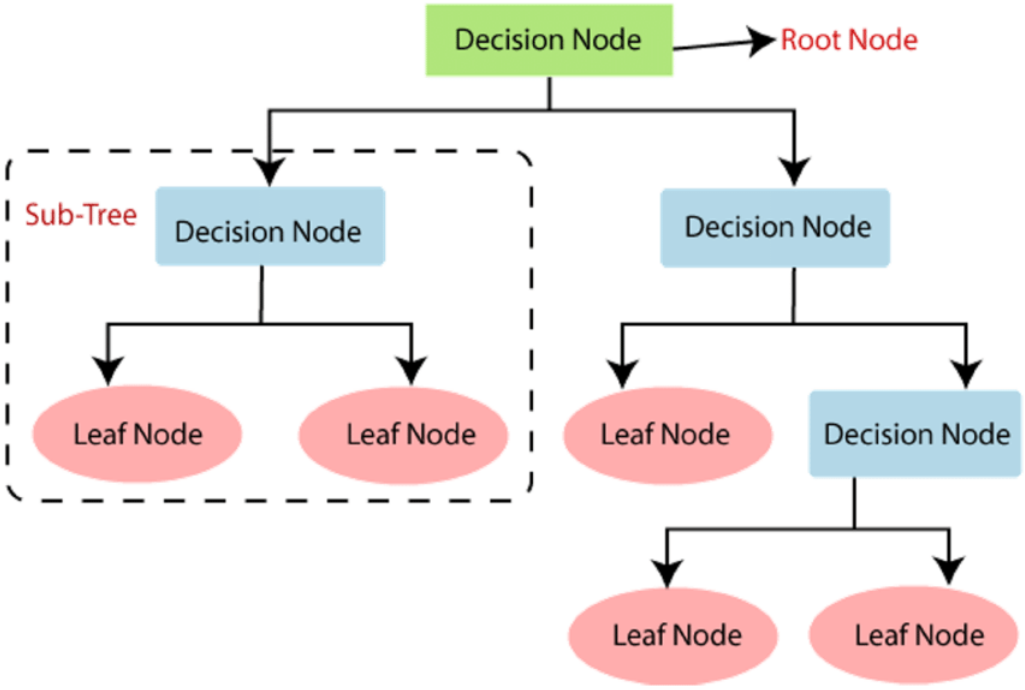


Figure 4.3: Classification tree important definitions

Source: Deshpande, Gite, and Aluvalu (2021)

4.3 Classification Trees

Classification trees, along with regression trees, are the two main types of decision trees. The former is used for classification problems and the latter for regression problems.

Figure 4.3 shows the format of decision trees. For decision trees, the following terms are used:

- **Root node**: Contains the entire sample of data from which the tree is followed depending on the decisions made.
- **Splitting**: Process of dividing a node into two or more sub-nodes
- **Decision node**: Represents a decision based on a single input variable that splits the data.
- **Leaf node**: Final nodes that are not split any further. With this, a classification is given for classification trees or a probability determined for regression trees. Also known as a *terminal node*.
- **Branch:** A sub-section or a tree.
- **Pruning:** Removing the sub-nodes of a decision node.
- **Parent node:** A node which divides into sub-nodes is known as the parent of the sub-nodes.
- **Child node:** The sub-nodes of a parent node are known as the child nodes of that node.

Decision trees were first introduced in 1963 by Morgan and Sonquist (1963). Classification trees then followed in 1972 when Messenger and Mandell (1972) split the data to maximise the sum of cases in the modal category. The predicted class was a mode.

No standardisation of variables is needed for any of the tree-based methods. Scaling the variables would only serve to similarly scale the cut-offs used for the splits without changing the eventual performance. It would also reduce the visual understandability of the trees.

At each node, the aim is to split the data into smaller, homogeneous groups. In this case, homogeneity means that most of the samples are from the same category. Consider a simple example with only two predictors seen in Figure 4.4, which allows easy visualisation on a 2D chart.

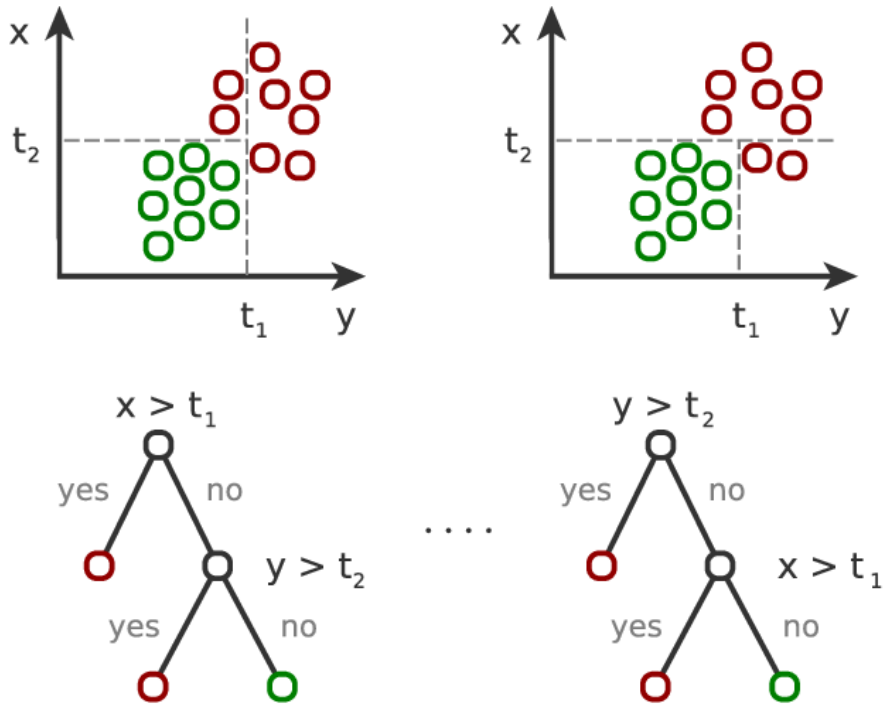


Figure 4.4: Classification trees splitting

Source: Horvát (2013)

A top-down approach is taken called *recursive binary splitting*. Start at the top of the tree with all observations, and then at each step, choose the split that results in the greatest homogeneity. This approach is considered greedy because it only considers the current step and not the one that could lead to the best performance in future steps. However, considering every single future step at every step of the process is not a viable approach, given the number of possibilities.

Splitting Criteria

Some measure of homogeneity is therefore needed. Three popular measures of this used for classification trees are

- **Gini index:** A measure of node impurity or variability within the terminal nodes.
- **Entropy:** Measure of entropy within terminal nodes.

- **Deviance:** Views the classification tree as a probability model and defines a measure related to the log-likelihood.

The deviance method will be used to determine the splits in this paper. It is defined as $-2 \times log(L)$ where L is the likelihood. The most probable model will be the one with the highest likelihood and, therefore, the lowest deviance. When choosing splits, the one that reduces the deviance by the most at each stage is chosen.

A major advantage of deviance over the other two measures is that it is weighted by the number of observations in each of the terminal nodes.

4.3.1 Pruning

The question becomes, at what stage should the splitting stop? If splitting goes on for too long, then the tree will be overfit to the data it was trained on and would not generalise well - consider the case where the splitting continues until every sample is correctly classified with many terminal nodes only containing one observation. Stopping when the splitting leads to some arbitrary reduction in deviance also leads to issues, as sometimes a small reduction in deviance in one step may lead to a large reduction in a future step.

Similar to regularisation, cost-complexity pruning (also known as weakest link pruning) introduces a penalty to the impurity measure, $\alpha|T|$ where $|T|$ is the number of terminal nodes, and α is a positive tuning parameter. The tuning parameter α, therefore, controls a trade-off between the complexity of a tree and its fit to the training data. This parameter α is determined using cross-validation.

4.3.2 Algorithm

Using the steps detailed in James et al. (2013) for regression trees, one for classification trees can be detailed as follows:

1. Build a large tree using recursive binary splitting, stopping only when each terminal node has fewer than some minimum number of observations
2. Apply cost complexity pruning to obtain a sequence of best subtrees as a function of α
3. Use K-fold cross-validation to choose α. Divide the training set into k folds and for each $k = 1,..,K$:
   (a) Repeat steps 1 and 2 on all but the k^{th} fold of the training data
   (b) Evaluate prediction error on the data in the k^{th} fold, as a function of α.
   (c) Average the results across all folds for each value of α and pick α to minimise the average error.
4. Return the subtree from step 2 that corresponds to the chosen α

4.3.3 Advantages/ Disadvantages

James et al. (2013) detailed the advantages and disadvantages of decision trees in general, which covers the points for classification trees:

Advantages

- Easy to explain and visualise - more so than even logistic regression.
- Decision trees are believed to more closely mirror human decision making than other classification approaches.

- Can handle qualitative predictors without the need for dummy variables. Scaling of variables is also not needed.

Disadvantages

- Prediction accuracy is not as high as some of the other models.
- Trees are not very robust - a small change in the data can cause a large change in the final estimated tree.

The relative performance of decision trees and linear models can be seen in Figure 4.5. This contains a classification example where the yellow and green regions indicate two different regions. In the top row, the true decision boundary is linear, and in the bottom, it is non-linear. The first column shows the performance of linear models in both these cases, and the second shows the performance of decision trees. As this diagram shows, the performance depends on what the true underlying model is, and there is, therefore, use for both these models.

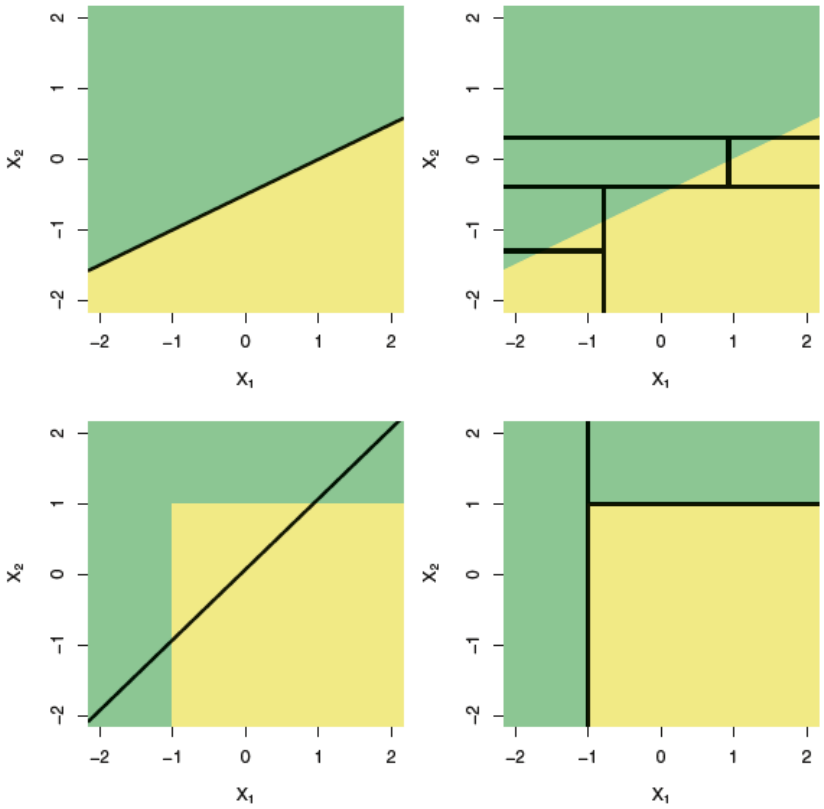


Figure 4.5: Tree vs. linear model classifications

Source: James et al. (2013)

4.4 Bagging/Bootstrap Aggregation

Decision trees tend to have high *variance*. A model is referred to as having high variance if the data could be split into two halves, and the resulting models produced from both halves would be vastly different. A model with high variance can also be referred to as one that is not very robust.

Bagging (also known as *bootstrap aggregation*) is the name of a general procedure for reducing the variance of statistical modelling. In this paper, bagging is referred to specifically in the context on decision trees. Bagging allows one to fit many trees by using repeated sampling

with replacement. For each tree, a random sample is taken using sampling with replacement, and an unpruned classification tree is fit based on that data. To get predictions for each model point, it is run through each tree. For categorical response variables, like the one here, the majority decision amongst all the trees is the one chosen. This process can be seen in Figure 4.6.

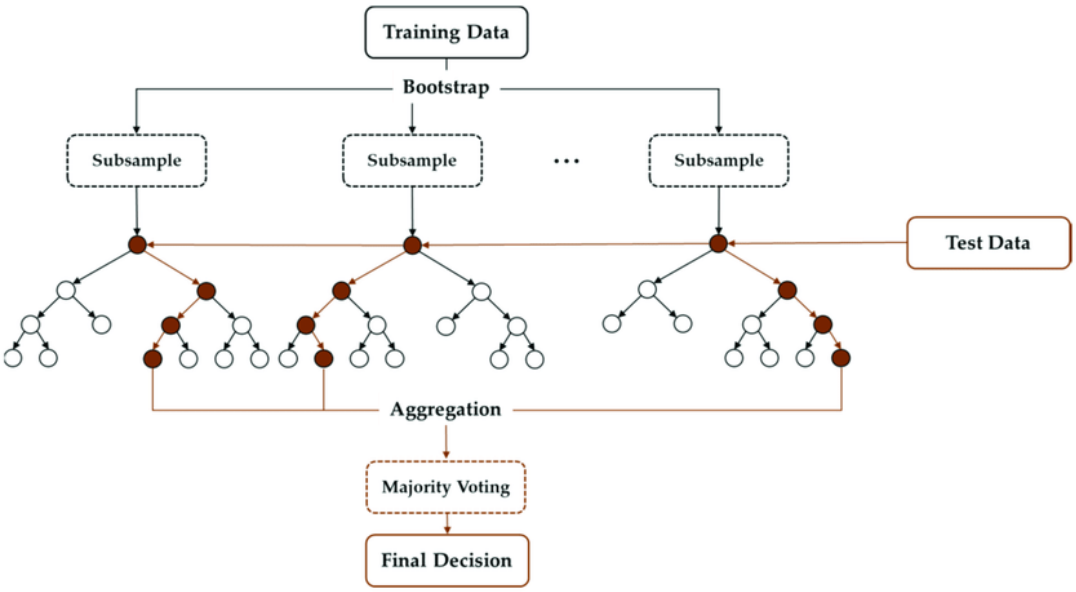


Figure 4.6: Bagging process explained

Source: Jeong et al. (2022)

James et al. (2013) explained that bagging works because it uses the principle that averaging a set of observations reduces variance. Given a set of n independent observations $Z_1, ..., Z_n$ each with a variance σ^2, the variance of the mean $\overline{Z}$ is given by $\frac{\sigma}{n}$. Of course, in this case, there is not access to many independent training sets. However, using bootstrap methods, an amount B repeated samples can be taken from a (single) training data set, which generates B bootstrapped training sets. B is the chosen number of trees to average over in this case.

4.4.1 Out-of-Bag Error

For bagging, cross-validation is not needed. The test error is estimated by using out-of-bag (OOB) errors. OOB basically uses the unused observations in each tree in their respective trees to get estimates. These estimates are then averaged over all trees where that observation was an OOB observation. From this, error rates for each observation are calculated. Since these error rates are based on out-of-sample observations, they serve as an estimate of the test error, just like cross-validation errors. According to James et al. (2013), it can be shown that each bagged tree makes use of around two-thirds of the observations. The remaining third will be the OOB sample. Each observation is, therefore, in an OOB sample roughly one-third of the time and thus will have around $\frac{B}{3}$ predictions each where B is the number of trees used.

To determine the number of trees, a plot of the OOB error vs. the number of trees can be generated, and if the OOB becomes quite stable, then the number of trees is sufficient. A large value for B will not cause the model to be overfit. For this model, the number of trees chosen will be 1000. Stability in the OOB error vs. the number of trees will determine if this is enough.

4.4.2 Variable Importance Measures

The easy visual interpretation of the trees is no longer an option since many trees are used. This takes away a major advantage of decision trees. However, a new concept of *variable importance* can be introduced by looking at the amount that a given predictor reduced the Gini index averaged over all B trees. This gives an idea of the mean decrease in the Gini index for each variable. The higher the average reduction, the more important the variable is seen as being to the model.

4.4.3 Advantages/ Disadvantages

For bagging, the advantages and disadvantages are the following:

Advantages

- Reduces variance found with decision trees.
- Improved accuracy over decision trees.
- Variable importance measures can be used to assess the importance of variables.

Disadvantages

- No longer has the simple visual interpretability that decision trees have.
- More complicated than decision trees.
- Slower than decision trees due to the number of decision trees that are created.

4.5 Random Forest

Bagging reduces the sampling variability of predictions by averaging over many trees. However, if the trees are highly correlated, the improvement will not be substantial. For a correlated set of random variables $Z_1, ..., Z_n$ with common variance σ^2 the variance of the mean $\overline{Z}$ is given by $\frac{\sigma}{n} + \boldsymbol{\frac{2\sigma^2}{n^2}}$

If there is one very strong predictor, most of the bagged trees will use this predictor. Random forests (Ho (1995); Breiman (2001)) decorrelate the trees by adding the additional rule that each time a split is considered, a random sample of $m \leq p$ predictors are chosen as split candidates where emphp is the total number of predictors. Bagging is, therefore, the special case where $m = p$. For a given tree, therefore, each split has to be one of those m predictors chosen for that tree. For each tree, a new set of m predictors is chosen. On average, therefore, $\frac{p-m}{p}$ of the trees will not even consider the strong predictor. This has the effect of decorrelating the trees so that the average predictions are less variable and, therefore, more reliable.

Here again, 1000 trees will be used. It is the norm, and therefore the default in the packages used, to set $m \approx \sqrt{p}$. Again, if the OOB error has relatively stabilised by this point, then it will show that the number of trees is enough. Again, there is no risk of causing overfitting by using too many trees. From repeating the models using all possible values of m, the value of m that results in the lowest OOB error will be chosen. Variable importance measures can be determined in the same way as for bagging, except that this time, the total Gini reduction is only divided by the number of trees that that variable was used in.

4.5.1 Advantages/ Disadvantages

Bagging is basically a special case of random forest. The advantages and disadvantages of random forest are therefore very similar with the following additions:

Advantages

- Reduces the variance found with bagging, making it more reliable.

Disadvantages

- Slightly slower than bagging as a new set of predictors must be chosen for each tree. Also slower if the value for m must be selected as well.

4.6 Boosting

Like bagging, boosting is a general term, but for this paper, it will be kept in the context of decision trees. Boosting refers to algorithms where base models sequentially learn from the mistakes made by the previous ones, thereby bringing many weak learners together into stronger learners. This differs from bagging and random forest, where all trees are grown independently.

The differences in the way that bagging and boosting models work can be seen in Figure 4.7. Bagging models work in parallel and independently, whereas information from one tree in boosting is used in the next one.

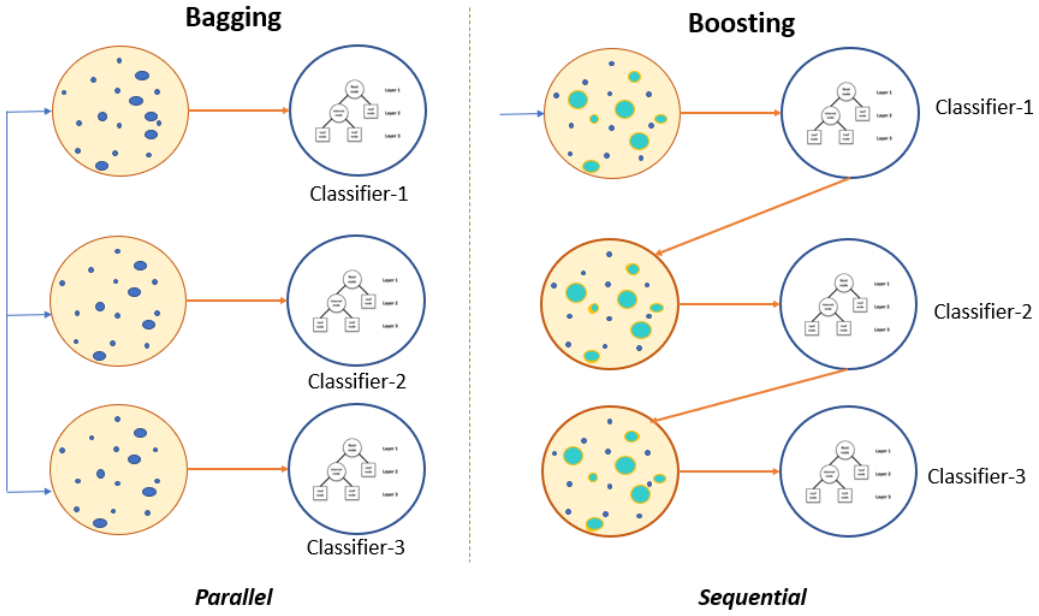


Figure 4.7: Bagging vs. boosting

Source: Singhal (2020)

4.6.1 Algorithm

The specific version of boosting used in this paper is known as *gradient boosting*. The basic algorithm for this is as follows:

1. A tree with a small number of splits, say d, is grown.
2. The residuals on this tree are then treated as the response variable for another tree also grown with d splits.

3. The residuals for this are then, in turn, used for another tree with d splits and so on.
4. The contribution of each tree is weighted down by a learning parameter, λ
5. In total, B trees are created, each one accounting for some more of the variation in y.
6. Initial trees capture the strongest patterns in the data and will contribute the most to the reduction in variance.

4.6.2 Hyperparameters

In general, methods that learn slowly tend to perform well. There are three main hyperparameters that control this that James et al. (2013) detailed as follows:

- The number of trees, B. Unlike bagging and random forest, boosting can overfit if B is too large, although this overfitting tends to occur slowly, if at all. Cross-validation is used to select B.
- Learning rate, λ, a small positive number which controls the rate at which boosting learns.
- The number d of splits in each tree which controls the complexity of the boosted ensemble.

4.6.3 Partial Dependence Plots

Variable importance plots show which features are most important for predicting a response but do not indicate how the predictors influence the predictions. Partial dependence plots seek to answer this. A partial dependence plot (PDP) can be created for each parameter. In order to create a PDP for a given parameter i, the following steps must be followed:

1. Replace the values of i in all observations to some value j.
2. Feed each observation into the chosen model and evaluate.
3. The most common classification over all observations will be the classification chosen for that combination of i and j.
4. Vary j over all possible values or a range of values, and the results will give a partial dependence plot for parameter i.

This essentially shows how one predictor influences the predictions given a fixed set of values for all other predictors. With PDPs, it is the shape of the trend and relative ranges between plots that are of importance. The absolute values are not important.

4.6.4 Extreme Gradient Boosting

Extreme gradient boosting (called *XGBoost* for short) is an efficient open-source implementation of the gradient boosting algorithm introduced by Chen et al. (2015). It contains several advanced techniques to improve the performance and speed of the algorithm. Chen and Guestrin (2016) reported that shortly after its release, it became very popular in competitive data science competitions, and it was the model that led to the most winning solutions in major competitions in 2015. As such, this will be the implementation used in the paper.

4.6.5 Advantages/ Disadvantages

For boosting in general, the advantages and disadvantages are the following:

Advantages

- Can handle non-linear relationships as it is based on a combination of decision trees.
- Good performance: The combination of multiple weak learners to form a strong model can improve performance compared to individual weak models.

Disadvantages

- Sensitive to overfitting, particularly if the number of weak learners is too large.
- Sensitive to outliers as every classifier tries to fix errors in predecessors.
- Reduced interpretability compared to single decision trees.

4.7 Support Vector Machines

At a basic level, support vector machines (SVMs) work by creating a line or hyperplane that separates the data into classes. Sometimes, there is a line or hyperplane that can perfectly split the data, and sometimes no such line exists. When such a line exists, the data is referred to as being linearly separable. When the data is not linearly separated, then sometimes there is a transformation that can be made to the data that will make it linearly separable. In other cases, a best-performing hyperplane can be used where some of the observations are incorrectly classified. The various possibilities will be explored further here.

4.7.1 Linearly Separable

Consider first the case where there are two features and two possible classifications which are linearly separable. Linearly separable means that there is a clear line that can be drawn between the classes, and everything to one side of the line belongs to one class, and everything to the other side belongs to the other class.

Figure 4.8 shows an example of a linearly separable set of observations where a clear line can be drawn between the two sets of observations. With linearly separable data, however, there exists an infinite number of these lines (or hyperplanes in higher dimensions) that can be drawn that perfectly separate the data. The question then becomes which line to use.

James et al. (2013) explained that a logical choice for this is known as the *maximal marginal hyperplane* (also known as the *optimal separating hyperplane*). For this, the perpendicular distance is calculated for each observation to a given separating hyperplane. The minimum distance from all observations to a given hyperplane is known as the *margin*. The maximal margin hyperplane is the separating hyperplane for which the margin is the largest. This hyperplane is, therefore, the one that has the largest minimum distance to all observations. The hope is that a large margin on the training data will have a large margin on the test data and, hence, classify the test data correctly. Figure 4.9 shows the maximal marginal hyperplane with the solid line. The margin is the distance from the solid line to either of the dashed lines.

This method, though, is quite sensitive as a single observation can change the choice of hyperplane dramatically. This can be seen in Figure 4.10.

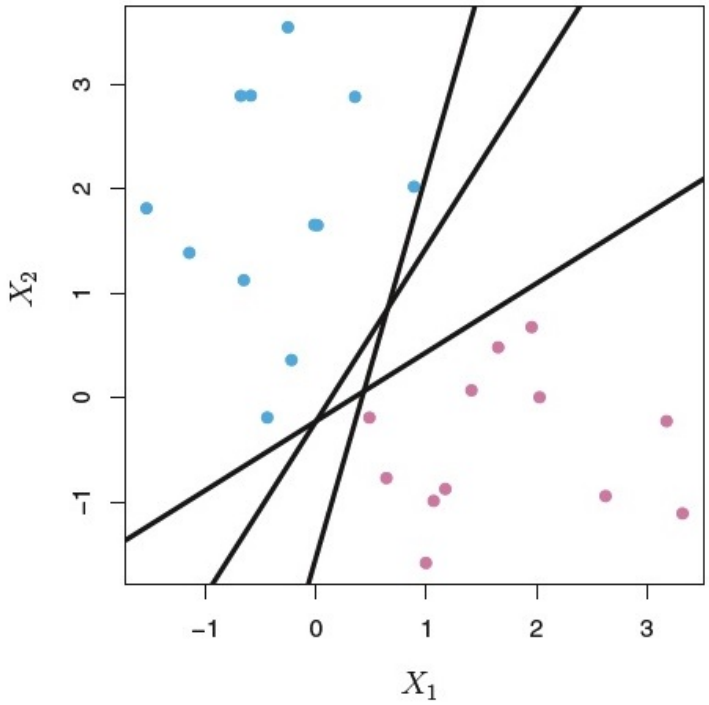


Figure 4.8: SVM linear separability

Source: James et al. (2013)

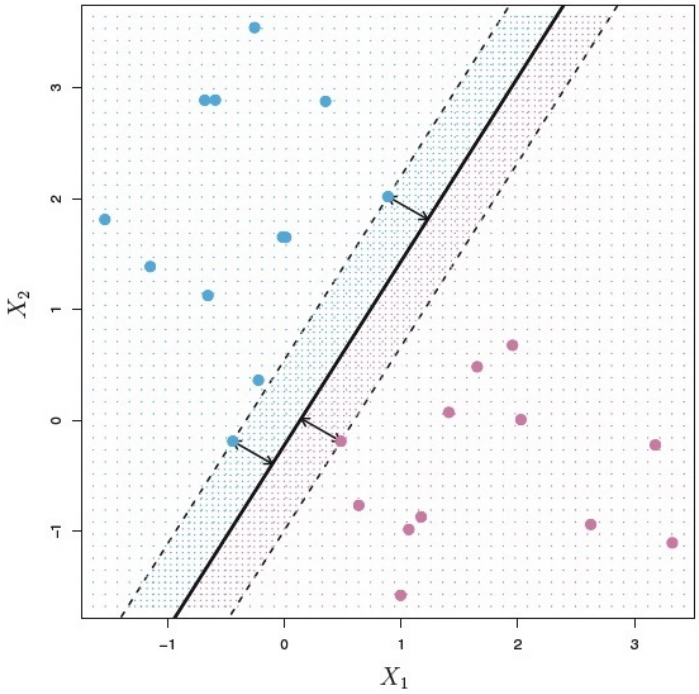


Figure 4.9: SVM maximal marginal hyperplane

Source: James et al. (2013)

4.7.2 Non-Separable

There are three hyperparameters that can be controlled in this case:

- **Cost:** How tolerant the hyperplane is to misclassification in the training set
- **Gamma:** How much influence single points have.

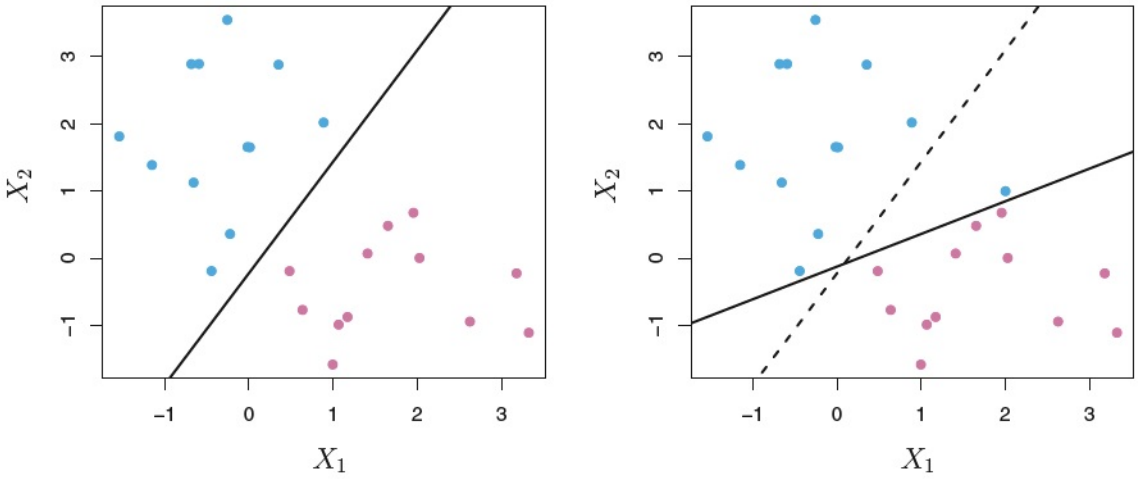


Figure 4.10: SVM maximal marginal hyperplane sensitivity

Source: James et al. (2013)

- **Kernel:** Transformation applied to the data.

Cost

Sometimes, the data is not easily separable, and it could be worthwhile to misclassify a few training observations in order to do a better job of classifying the remaining observations.

Values that are on the wrong side of their margin come at a *cost*. To understand the concept of cost, an understand of the terms emphslack variables, emphmargin construction and emphsupport vectors are needed. Er (2020) explained these terms as follows:

Slack Variables

Define slack variables $\xi = (\xi_1, \xi_2, ...\xi_n) \geq 0$ such that:

- If a data point is correctly classified and outside or on the margin, then $\xi_i = 0$.
- If a data point is correctly classified but within the margin, then $0 < \xi_i \leq 1$.
- If a data point is incorrectly classified, then $\xi \geq 1$ regardless of whether they are within the margin.

The goal is to maximise the margin while softly penalising points that lie on the wrong side of the margin boundary.

Margin Construction

Margin is maximised subject to a total budget of $\Sigma\xi_i^* \leq V$. V is the amount that the margin can be violated by the n observations. If $V = 0$, then no violations are allowed. This would only work if the data was linearly separable and would give the maximal marginal hyperplane. If $V \geq 0$, then at most V observations can be on the wrong side of the hyperplane. This is the case because if an observation is on the wrong side of the hyperplane, then for that observation, it must be that $\xi \geq 1$. As V increases, the margins increase because the model is more tolerant of violations to the margin. Similarly, as V decreases, margins narrow because the model is less tolerant of violations to the margin. V is chosen via cross-validation. Construction of the margin is, therefore, an optimisation problem. This is usually done using some sort of software.

Support Vectors

Support vectors are observations that lie directly on the margin or on the wrong side of the

margin for their class. Support vectors affect the support vector classifier. Observations that are on the correct side of the margin do not affect the support vector classifier. The more support vectors there are, the better the generalisation of the boundary. The number of support vectors is controlled by V. When V is large, the margin is wide, and many observations violate the margin, and there are, therefore, many support vectors. There are many observations that are involved in determining the hyperplane in this case.

Kernels

Sometimes, data may not be linearly separable, but a simple transformation may have the data linearly separable. Consider the case of Figure 4.11. Here, there is data that is not linearly separable, but it seems like some sort of circular separator would work. The transformation depicted in the picture makes it linearly separable.

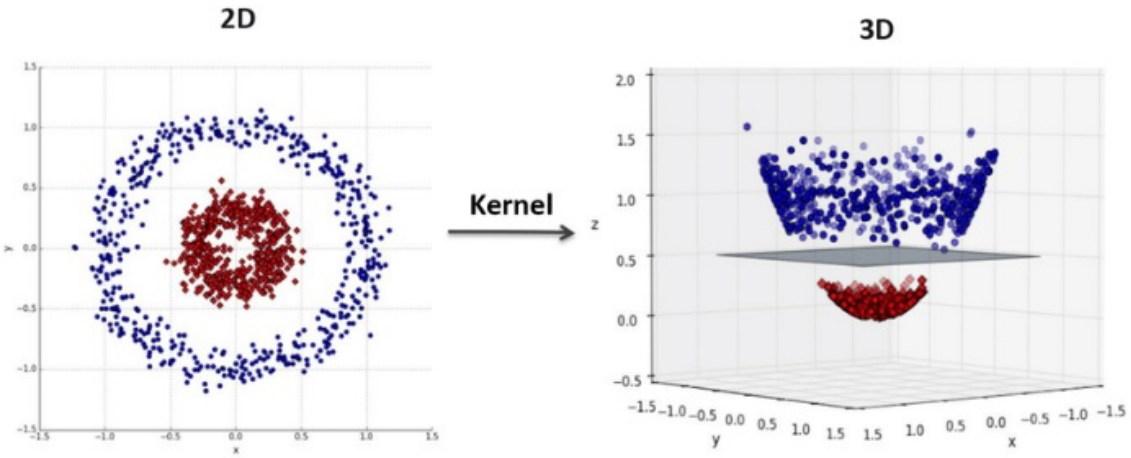


Figure 4.11: SVM kernel transformations to make data linearly separable

Source: Hachimi et al. (2020)

Kernel functions transform data so that a non-linear separator is able to transform to a linear separator in a higher number of dimension spaces.

Gamma

Specific to the radial kernel, gamma defines how far the influence of a single training point reaches. Gamma can take on any positive value. If the value of gamma is high, then the decision boundary will depend on points close to the decision boundary, and nearer points carry more weight than far away points. A large value of gamma can, therefore, similarly lead to overfitting.

4.7.3 Advantages/ Disadvantages

For SVM, the advantages and disadvantages are the following:

Advantages

- Works well when there is a clear margin between classes.
- Effective in high dimensional spaces.
- Relatively memory efficient.

Disadvantages

- Not good for large data sets.

- Does not perform well when target classes are overlapping.
- Not a probabilistic model, so it cannot explain the classification in terms of probability.

4.8 Neural Networks

Neural networks are a set of algorithms modelled loosely after the human brain that are designed to recognise patterns. There are various types of neural networks. Neural networks can be used for a variety of problems, such as prediction, classification, image analysis, and pattern recognition. This paper will focus on artificial neural networks (ANN), which are ideal for classification problems.

ANNs are organised into layers of nodes, and data moves through them in one direction. An individual node is connected to all nodes in the layer before it. To each of the incoming connections, a node will assign a number known as a *weight*. Each node will add the resulting product of each of the weights and values from the previous layer. Each node after the initial layer also has an activation function, which will do some sort of transformation to the sum of the weights. Hidden layers are those layers after the initial layer and before the final decision layer. Figure 4.12 depicts a simple ANN where there are no activation functions.

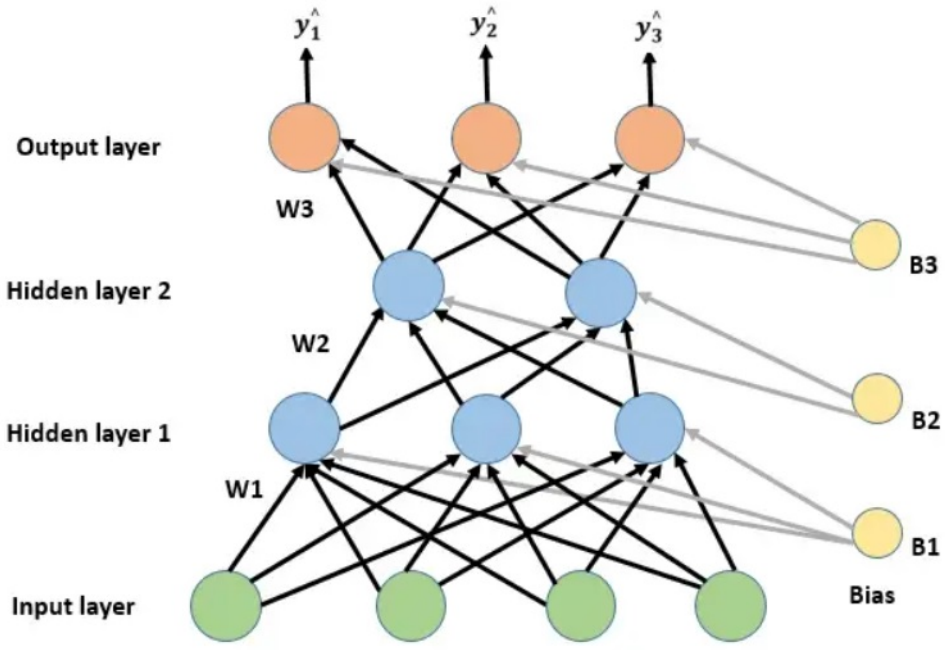


Figure 4.12: Artificial neural network with no activation functions

Source: Luhaniwal (2019)

Essentially, the output can be presented as

$$\hat{y_1} = \boldsymbol{W3}(\boldsymbol{W2}(\boldsymbol{W1} \times \boldsymbol{X} + B1) + B2) + B3 = \boldsymbol{W} \times \boldsymbol{X} + \boldsymbol{B}$$

where:

$$\mathbf{W} = \mathbf{W1} \times \mathbf{W2} \times \mathbf{W3}$$

This is simply a linear transformation of weights and inputs and is, therefore, similar to the linear regression model. It can, therefore, only handle linear relationships between variables. Activation functions address this by performing a transformation to the weighted sum, which becomes the new output from a node. There are various types of activation functions. These

are looked at in Section 4.8.1, including how to deal with the case where the response is a categorical variable.

Another question that arises is how to deal with the case where some of the input variables are categorical. Consider again an example where the input variable is car colour, which could be either red, blue or green. This could then be converted using the code red = 0, blue = 1, green = 2, for example. The issue with this, though, is that there is no natural ordering to different car colours, so the order of the numbers would be arbitrary. In this case, a technique called *one-hot encoding* is used. Zheng and Casari (2018) detailed this as each bit representing a possible category. If the variable cannot belong to multiple categories at once, then only one bit in the group can be "on". In other words, in the car colour example, these colours could be relabelled as red = [1,0,0], blue = [0,1,0] and green = [0,0,1]. In neural network terms, these then become essentially three different input nodes, with one of them having a value of 1 and the other two having a value of 0, depending on the actual car colour.

4.8.1 Activation functions

Back to the neuron analogy, activation functions process the input signals and return an output signal. For ANNs, all inputs in the same layer will have the same activation function. Figure 4.13 depicts some of the most common types of activation functions. For multiclass classification problems such as this, it is most common to use the softmax activation function in the output later because it returns values between 0 and 1, with the most likely class having the higher values.

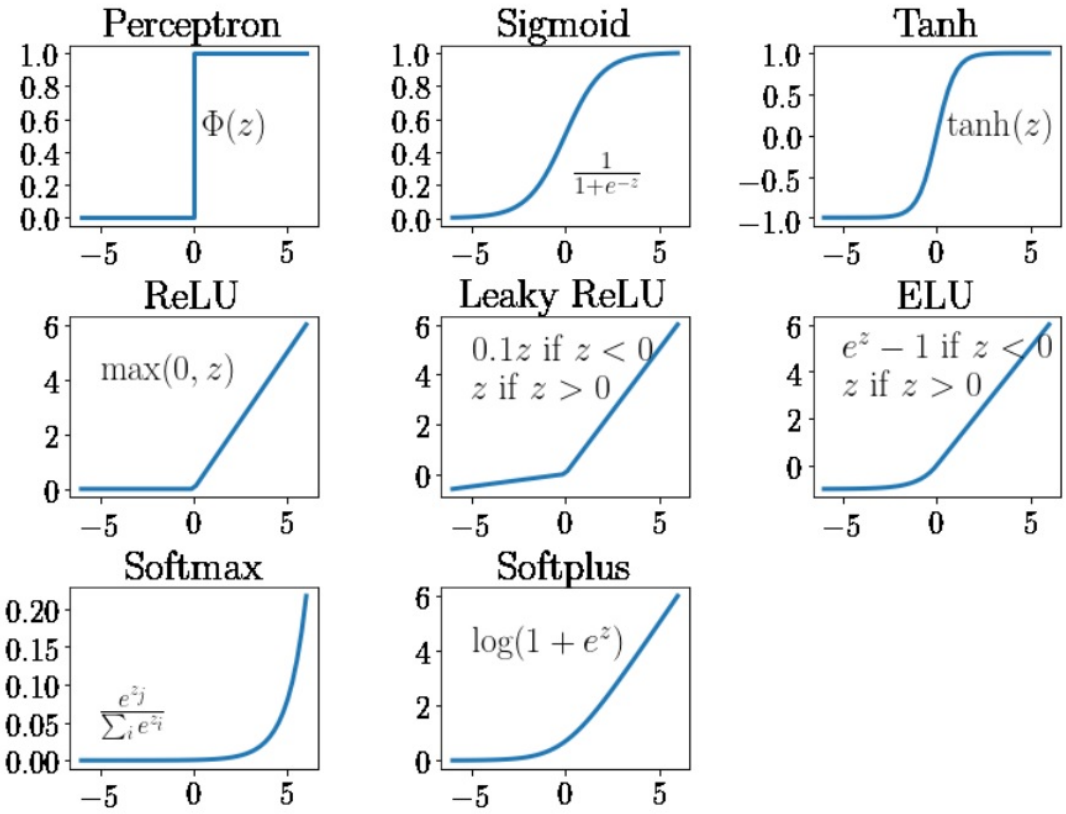


Figure 4.13: Activation functions

Source: Johnson et al. (2020)

4.8.2 Loss Function

The loss function is simply a measure of the "correctness" of a model that can be used to compare models. During the training of neural networks, parameters are updated to minimise

the loss function. There are various loss functions that can be used, but for classification problems such as this one, *cross-entropy* is typically used.

Cross-entropy measures the difference between two probability distributions for a given random variable. It can be represented mathematically as

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

where:

$\mathcal{X}$ is the space of possible outcomes,

$p(x)$ is the actual probability, and

$q(x)$ is the predicted probability distribution.

Goodfellow, Bengio, and Courville (2016) explained that for binary classification problems, the sigmoid activation function is used. It ensures the output values lie between 0 and 1, which mimic probability values.

4.8.3 Selecting Weights

An optimisation algorithm is needed that determines how weights are optimised to reduce the loss function. There are various algorithms that can be used. One of the simplest, and the one that will be used here, is *gradient descent*. The gradient descent algorithm at a basic level can be described as follows:

1. Assign initial random values for network weights.
2. Present input values to the network.
3. Calculate the error using the loss function
   $J(W) = -\frac{1}{2n}\sum_{i=1}^{n}[y^{(i)}log(\hat{y}^{(i)}) + (1 - y^{(i)})log(1 - \hat{y}^{(i)})]$.
4. Update the weights according to formula
   $w_j := w_j - \alpha\frac{\partial}{\partial w_j}J(w)$
   where α is the learning rate and $0 \leq \alpha \leq 1$.
5. Continue iterating until *convergence*.

In the above, the goal is to minimise $J(w)$ with respect to w_j. In the weight update function, partial derivatives give us the slope of $J(w)$ at point w. Each round of feeding all the inputs before calculating the cost function and derivatives is known as an *epoch*. α, which is the learning rate, determines the amount that the weights are adjusted after each epoch.

This continues until there is convergence. Convergence in this context basically means that the training error and model parameters have reached a stable state after multiple training iterations. Stable here could be defined as the changes between steps being less than a specified value. Once a model reaches convergence, it means that the optimal values have been found for the parameters that best fit the training data and that further training is unlikely to improve the performance of the model. If the learning rate, α, is too small, then it will take a really long time to converge. On the other hand, if α is too big, optimal values could be missed, and the model will fail to converge.

4.8.4 Batch Size

With batch gradient descent learning, the weights are updated based on all samples in the training set. An alternative to this is *stochastic gradient descent*, where weights are updated after each observation is processed instead of using the sum of all accumulated errors over all samples. Here, an epoch contains only one input. A compromise between the batch and stochastic gradient descents is known as *mini-batch learning*. Here, batches of smaller sizes for each epoch are chosen.

4.8.5 Regularisation

Srivastava et al. (2014) explained that neural networks contain multiple non-linear hidden layers, allowing them to capture a wide range of complex patterns and relationships. However, with limited training data, many of these complicated relationships and patterns will result from sampling noise and may not exist in the test data, even if they are drawn from the same distribution as the training data. This leads to overfitting.

There are various regularisation techniques that aim to prevent this overfitting. This paper uses the following: early stopping, drop-out and weight decay.

Early Stopping

The training error tends to decrease as the number of iterations increases as the parameters adjust to better capture the data. The model tends to overfit the training data if not stopped soon enough. After a certain point, the error on the validation set will increase. This can be seen in Figure 4.14 (Gençay and Qi, 2001).

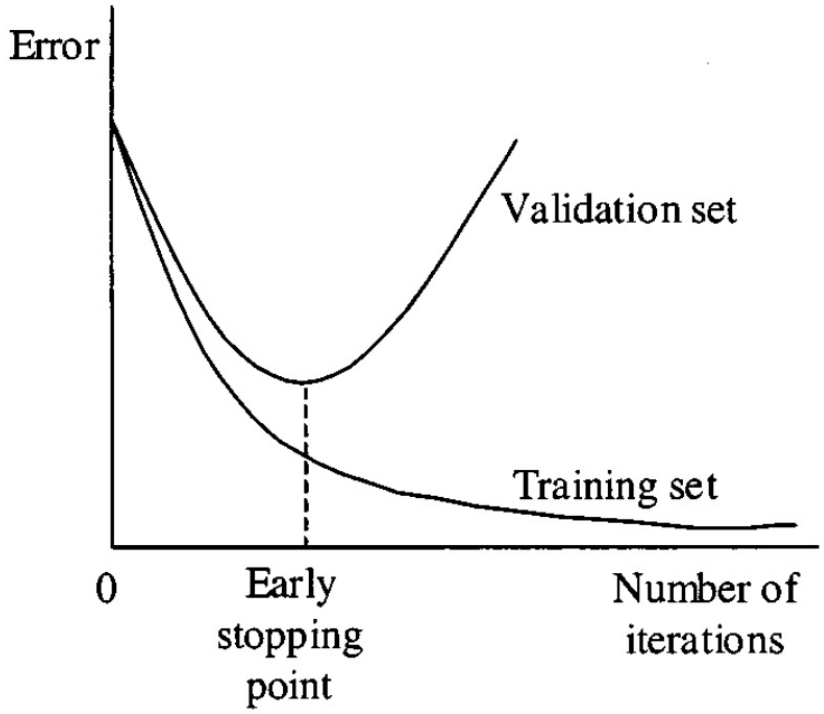


Figure 4.14: Neural networks early stopping

Source: Gençay and Qi (2001)

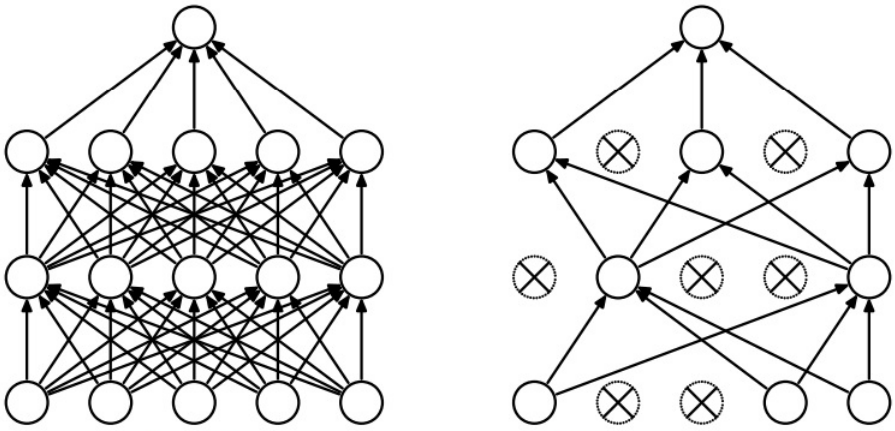
One possible approach is to hold out a part of the training data and use it as a validation set. The error measure on this validation set measured after each epoch often shows a decrease at first and then an increase as the model starts to overfit. Training can be stopped at the point of smallest error with respect to the validation data set (Bishop and Nasrabadi, 2006).

In Figure 4.14, everything to the left of the early stopping point the model will be underfitting, and to the right, the model will be overfitting.

Drop-out

Model combinations almost always improve the performance of machine learning models. For neural networks, the different models can be simulated by using the drop-out technique. Here, the term *drop-out* refers to dropping out nodes in a neural network. The simplest version of this would be retaining each node with a fixed probability, p, which is independent of the other nodes. p itself can be chosen using a validation set or set at 0.5, which tends to be close to optimal for a wide range of networks. For input nodes, though, the optimal probability of retention is closer to 1 than 0. If a unit is retained with probability p during training, then the outgoing weights of that unit are multiplied by p when the test set is used (Srivastava et al., 2014).

Figure 4.15 depicts an example of a neural network before and after applying Drop-out. On the left is a standard neural network with two hidden layers. The image on the right is the same neural network after applying drop-out where the crossed-out units have been dropped.

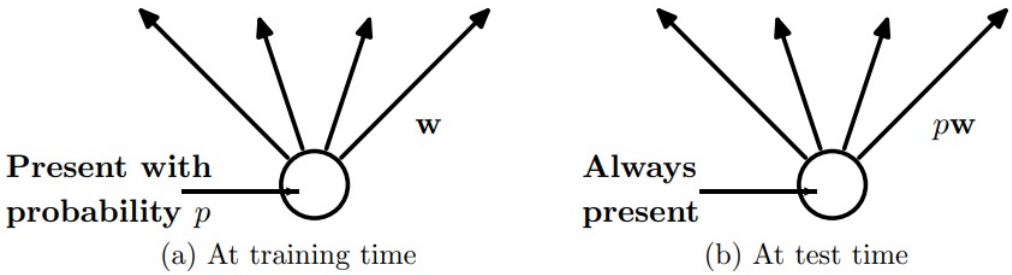


(a) Standard Neural Net (b) After applying dropout.

Figure 4.15: Neural networks drop-out

Source: Srivastava et al. (2014)

Figure 4.16 then shows how a node with weight w and probability p is then represented as a node with probability $p \times w$ at test time. Therefore, the output at the test time is the same as the expected output at training time.



(a) At training time (b) At test time

Figure 4.16: Neural networks drop-out probabilities

Source: Srivastava et al. (2014)

Weight Decay

Weight decay is a regularisation technique that can be used to prevent overfitting in neural networks. It works by adding a term to the loss function that penalises large weight values. Ridge regression (L^2 parameter norm penalty) is commonly used. A hyperparameter, α, controls the extent of regularisation by controlling the contribution of the penalty term. Krogh and Hertz (1991) explained that weight decay works for the following two reasons:

1. It suppresses any irrelevant components of the weight vector by choosing the smallest vector that solves the problem.
2. It can suppress some of the effects of static noise on the targets if the size is chosen correctly.

4.8.6 Advantages/ Disadvantages

For neural networks, the advantages and disadvantages are the following:

Advantages

- Can handle complex and non-linear data.
- Generalise well. Great at inferring unseen relationships.

They are thus great when interpretable results are not needed. They work particularly well for image classification, such as when the need is just to determine if a picture is a cat or a dog, but there is no need to explain why such a classification was made.

Disadvantages

- Black boxes. The amount that each independent variable influences the dependent variables cannot be seen.
- Require large amounts of data, computational power and memory to train.
- Susceptible to overfitting and underfitting, although techniques were discussed here to address that.

4.9 Naive Bayes

Naive Bayes is a classifier based on Bayes' theorem.

$$Bayes\ Theorem: P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Theorem provides a formula for calculating conditional probabilities. It allows us to find the probability of an event A happening, given that an event B has occurred. This can be extended for a number of variables.

$$P(Y = y|X = (x_1, ...x_n)) = \frac{P(X = (x_1, x_2, ..., x_n)|Y = y)}{P(X = (x_1, x_2, ..., x_n))}$$

4.9.1 Independence of Features

If the features are independent of each other, this can then be simplified to the following:

$$P(Y = y|X = (x_1, ...x_n)) = \frac{P(X_1 = x_1|Y = y)P(X_2 = x_2|Y = y)...P(X_n = x_n|Y = y)}{P(X_1 = x_1)P(X_2 = x_2)...P(X_n = x_n)}$$

Indeed, naive Bayes does assume that all features are independent. In practice, this is usually not the case, which is where the term "naive" comes from. Nonetheless, in practice, the model does tend to perform quite well.

Figure 3.1 in Section 3.4.1 showed the correlations between the different features. Most variables are not very correlated, but there were a couple of higher correlation pairs. The first inn median and second inn median had a correlation score of 0.58, and the ground length and ground width pairs had a correlation score of 0.78. These could challenge the assumption of independence of features and lead to poorer results for naive Bayes compared to other models which make no such assumption.

4.9.2 Distribution of Features

When the features take up continuous values, it is assumed that these values are sampled from a Gaussian distribution. The conditional probability $P(X_i|Y)$ then becomes a probability density function and can be expressed as follows:

$$f(X_i = x_i|Y = y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2})$$

When the features are discrete, the estimation of $P(X_i|Y)$ is more straightforward, as is given by the following formula:

$$P(X_i = x_i|Y = y) = \frac{\#(X_i = x, Y = y)}{\#(Y = y)}$$

4.9.3 Laplace Smoothing

If there is an observation in a class in the test set that is not present in the training set in the i^{th} variable, for example, then the term $P(X_i = x_i|Y = y)$ will be 0. In the naive Bayes formula, this will always yield a probability of 0 overall.

Laplace smoothing, also called additive smoothing, is a technique used to avoid zero probabilities in naive Bayes. The formula for Laplace smoothing is as follows:

$$P(X_i = x_i|Y = y) = \frac{\#(X_i = x, Y = y) + \alpha}{\#(Y = y) + \alpha * K}$$

where:

α is the smoothing constant

K is the number of features in the data

In the test set, the teams Gujarat Titans and Lucknow Super Giants are present and are not in the training set, so this hyperparameter is important. By definition, all categories are in the training set, so there is no scope to use cross-validation to set this value. The norm is to set the value of α as 1, which is what will be used in this paper.

4.9.4 Advantages/ Disadvantages

For naive Bayes, the advantages and disadvantages are the following:

Advantages

- Easy to understand and implement.
- Assumption of independence makes algorithm very fast compared to complicated algorithms.

- Works well with high-dimensional data, i.e. where there are many features such as text classification and email spam detection.

Disadvantages

- Assumption of independence.
- When there is lots of data, complex models tend to perform better.

4.10 Conclusion

In this chapter, the methodology used in this paper was covered. First, there was a look at the modelling approach. This entailed a look at how the data was split, the type of models used, and the evaluation criteria used. Then, a range of models that will be used were explained in terms of how they work and their advantages and disadvantages. This chapter is the foundation for the next one, where all this is used to evaluate the various models and the best model chosen.

Chapter 5

Model Application

In this chapter, the results from all the models explored in the previous chapter are analysed. Along with this, the chosen hyperparameters and parameters are discussed for each model. Possible reasons that a given model may have performed well or poorly are discussed. Finally, a comparison is made of all the models, and the best model for the purpose of this paper is chosen.

5.1 Logistic Regression

The first model discussed is logistic regression.

5.1.1 Training of Model

Lasso regression was used to prevent overfitting. Figure 5.1 shows the values of all the coefficients and how these change as the lambda parameter changes in the Lasso regression. The vertical blue line depicts the log lambda corresponding to the minimum CV MSE. This blue line, therefore, also shows the coefficients that will be used. How this was calculated is shown in Figure 5.2.

Figure 5.2 shows the CV MSE at different lambda levels. The blue line here marks the minimum and is when the log lambda value is -6.17, which means the lambda value is 0.002. The dotted line marks off one standard deviation from this mean, which is sometimes used as the value for Lambda. The log lambda value there is -2.64, which corresponds to a lambda value of 0.07.

5.1.2 Final Model

Table 5.1 shows the weights used and the corresponding odds ratios. A negative value for the odds ratio implies an inverse relationship between the dependent and independent variables. In this context, it means that as the value of that feature increases the chance of the defending team avoiding defeat decreases.

Base categories are those that come first alphabetically. The base defending and chasing team is therefore Chennai Super Kings. The base defending stadium is away, the base defending toss is lost, and the base time is day.

Surprisingly, the largest odds ratio comes from the chasing team being Pune Warriors. The next highest value is more expected and comes in the form of the first innings score. This reinforces the point that every run counts.

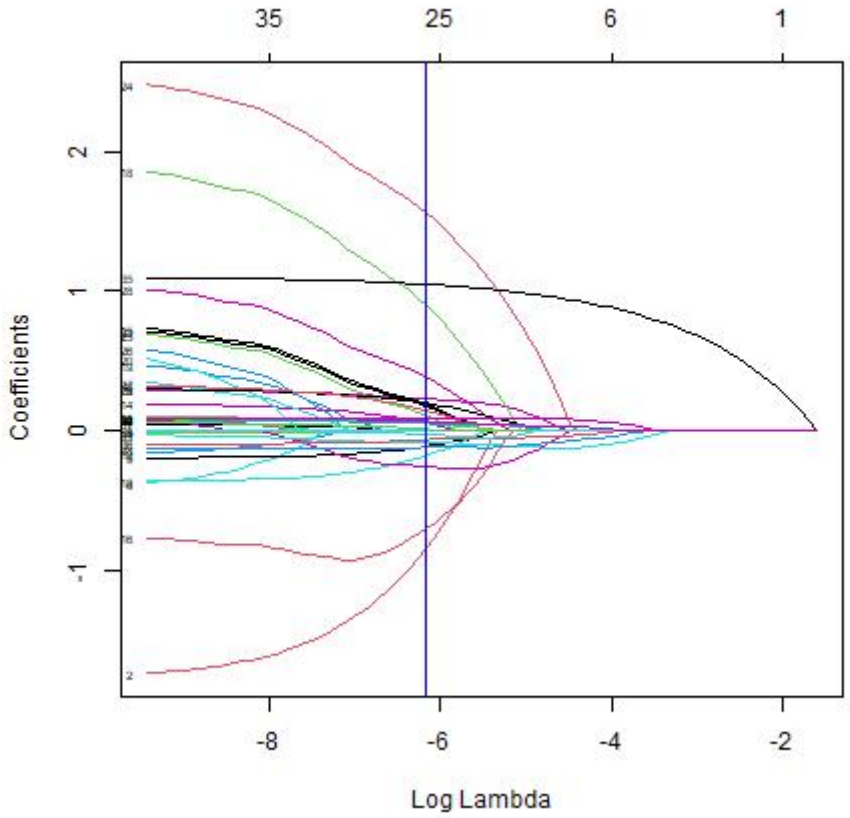


Figure 5.1: Logistic regression coefficients with Lasso regression

5.1.3 Results

The accuracy on the test set is 70.27%. The Brier score for the model is 0.2. The distribution of the accuracy can be seen in Figure 5.3. According to this, the model performs in line with its predicted probabilities. When it gives a higher probability of a particular outcome, it tends to get the predictions right more of the time. In matches where the model determines that the outcome could go either way, it seems to make the correct prediction roughly half of the time, which is in line with that.

5.2 Classification Tree

Tree models are looked at next, starting with classification trees.

5.2.1 Training of Model

To prevent overfitting for classification trees, pruning is performed. Pruning was done using cross-validation. Figure 5.4 shows for a given number of terminal nodes what the optimal alpha would be and the corresponding CV error. A minimum occurs when there are six terminal nodes. For this, the optimal alpha is 6.5.

5.2.2 Final Model

Figure 5.5 shows the final classification tree model. It is interesting to note that all splits had to do with either the first innings score or the name of the chasing team.

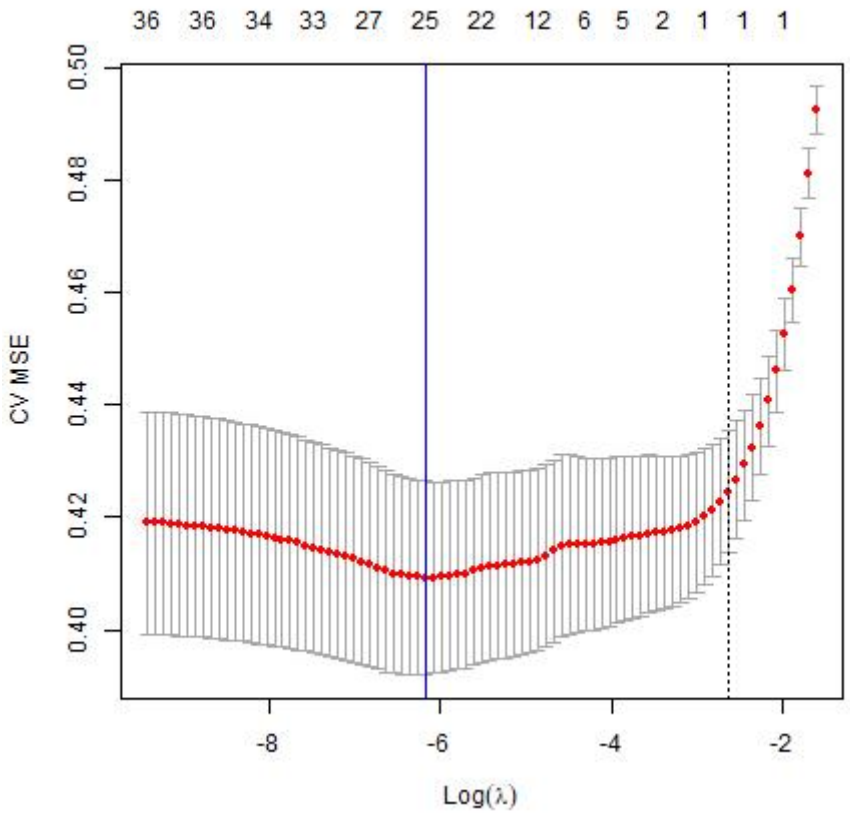


Figure 5.2: Lasso regression CV MSE vs. log lambda

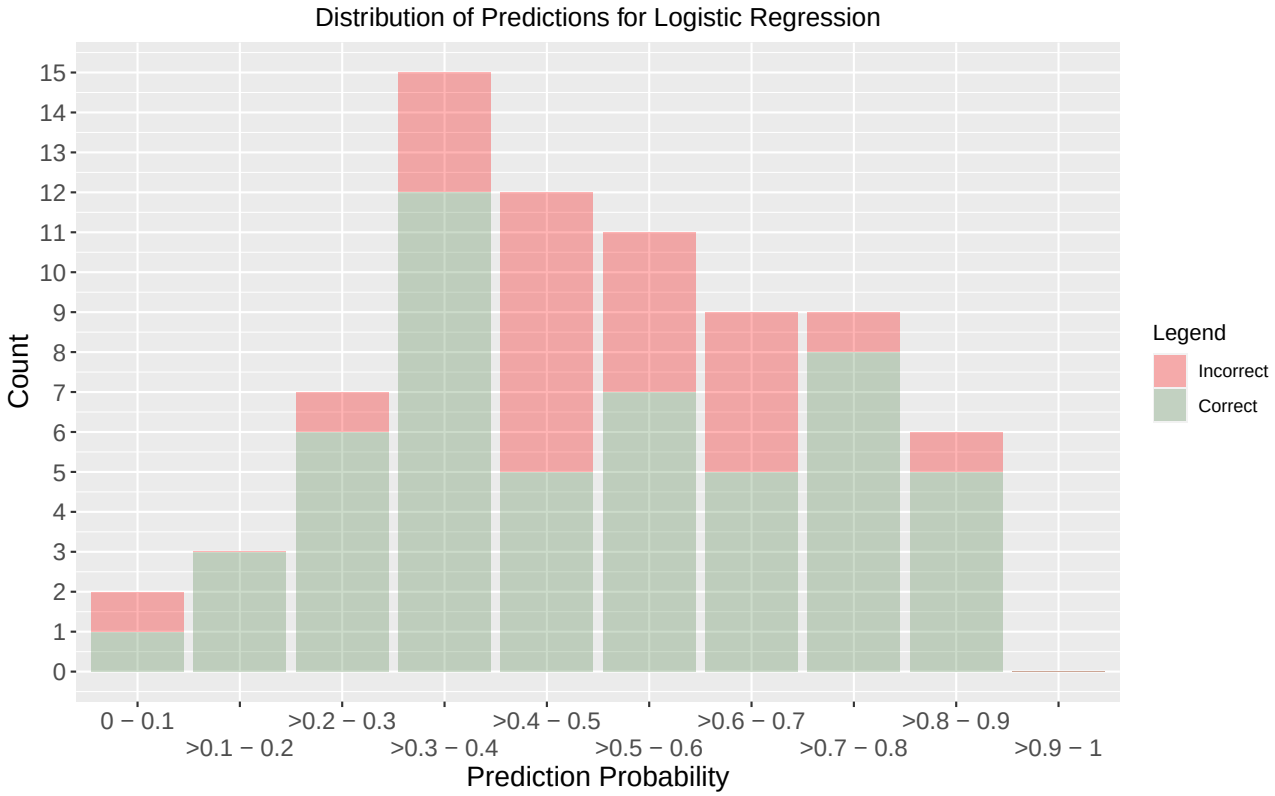


Figure 5.3: Logistic regression breakdown of accuracy in different probability ranges

	Coefficient	Odds Ratio
Intercept	0.54	-0.61
Defending Team: Deccan Chargers	1	0
Defending Team: Delhi Capitals	1	0
Defending Team: Gujarat Lions	0.43	-0.85
Defending Team: Gujarat Titans	1	0
Defending Team: Kochi Tuskers Kerala	1	0
Defending Team: Kolkata Knight Riders	1	0
Defending Team: Lucknow Super Giants	1	0
Defending Team: Mumbai Indians	1.26	0.23
Defending Team: Pune Warriors	1	0
Defending Team: Punjab Kings	0.89	-0.12
Defending Team: Rajasthan Royals	0.83	-0.18
Defending Team: Rising Pune Super Giants	1	0
Defending Team: Royal Challengers Bangalore	1	0
Defending Team: Sunrisers Hyderabad	1.06	0.06
Chasing Team: Deccan Chargers	2.46	0.9
Chasing Team: Delhi Capitals	1.21	0.19
Chasing Team: Gujarat Lions	0.49	-0.71
Chasing Team: Gujarat Titans	1	0
Chasing Team: Kochi Tuskers Kerala	1	0
Chasing Team: Kolkata Knight Riders	1	0
Chasing Team: Lucknow Super Giants	1	0
Chasing Team: Mumbai Indians	0.77	-0.26
Chasing Team: Pune Warriors	4.79	1.57
Chasing Team: Punjab Kings	1.2	0.18
Chasing Team: Rajasthan Royals	1	0
Chasing Team: Rising Pune Super Giants	1	0
Chasing Team: Royal Challengers Bangalore	1.11	0.11
Chasing Team: Sunrisers Hyderabad	1.47	0.39
Defending Stadium: Home	1.22	0.2
Defending Toss: Neutral	1.16	0.15
Time: Day/Night	1.06	0.05
Defending Toss: Won	1.05	0.05
Bat 2 Strength	0.92	-0.09
Bowl 2 Strength	1.09	0.09
First Inn Score	2.86	1.05
First Inn Median	0.94	-0.07
Second Inn Median	1.01	0.01
Ground Altitude	0.89	-0.12
Ground Length	0.97	-0.03
Ground Width	1.07	0.07

Table 5.1: Logistic regression coefficients and odds ratios

5.2.3 Results

The accuracy on the test set is 62.16%. With tree models, there are probabilities attached to final predictions. This allows the calculation of the Brier score and a breakdown of the accuracy for different prediction probabilities. The Brier score here is 0.22. For the classification tree here, the distribution of predictions can be seen in Figure 5.6.

The model does not perform as well as logistic regression. The model has a higher Brier score and a much lower accuracy. In addition, looking at Figure 5.6, there are far too many observations at the centre, meaning there are too many observations where the model is uncertain about the result. The model also gets these wrong more than would be expected by the predicted probabilities.

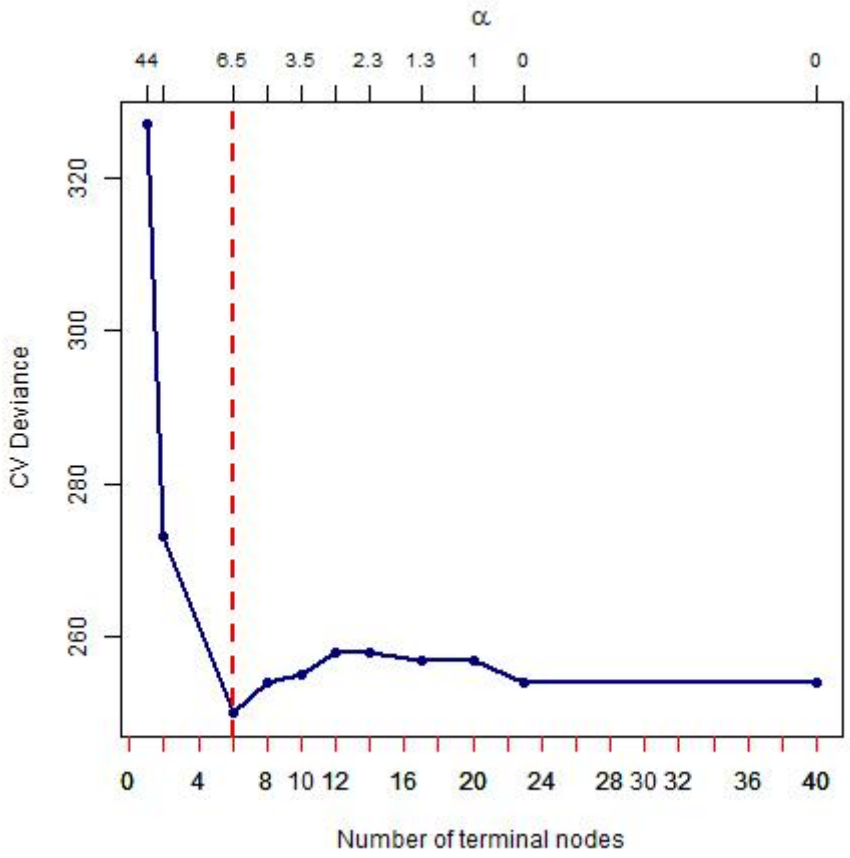


Figure 5.4: Bagging CV error vs number of terminal nodes

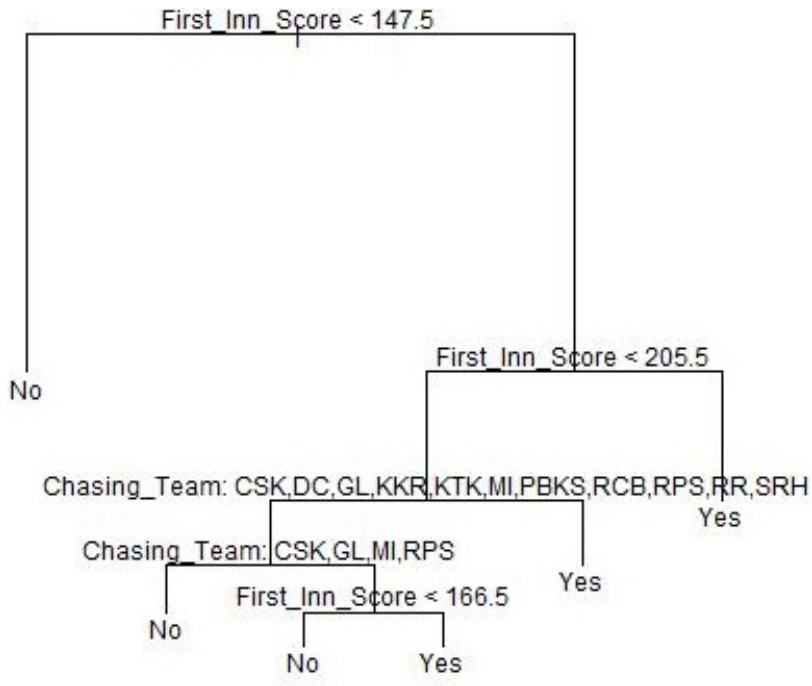


Figure 5.5: Classification tree final model

When it comes to classification trees, however, this model is quite basic, and the next few models offer various improvements to this model. In theory, these models should perform better.

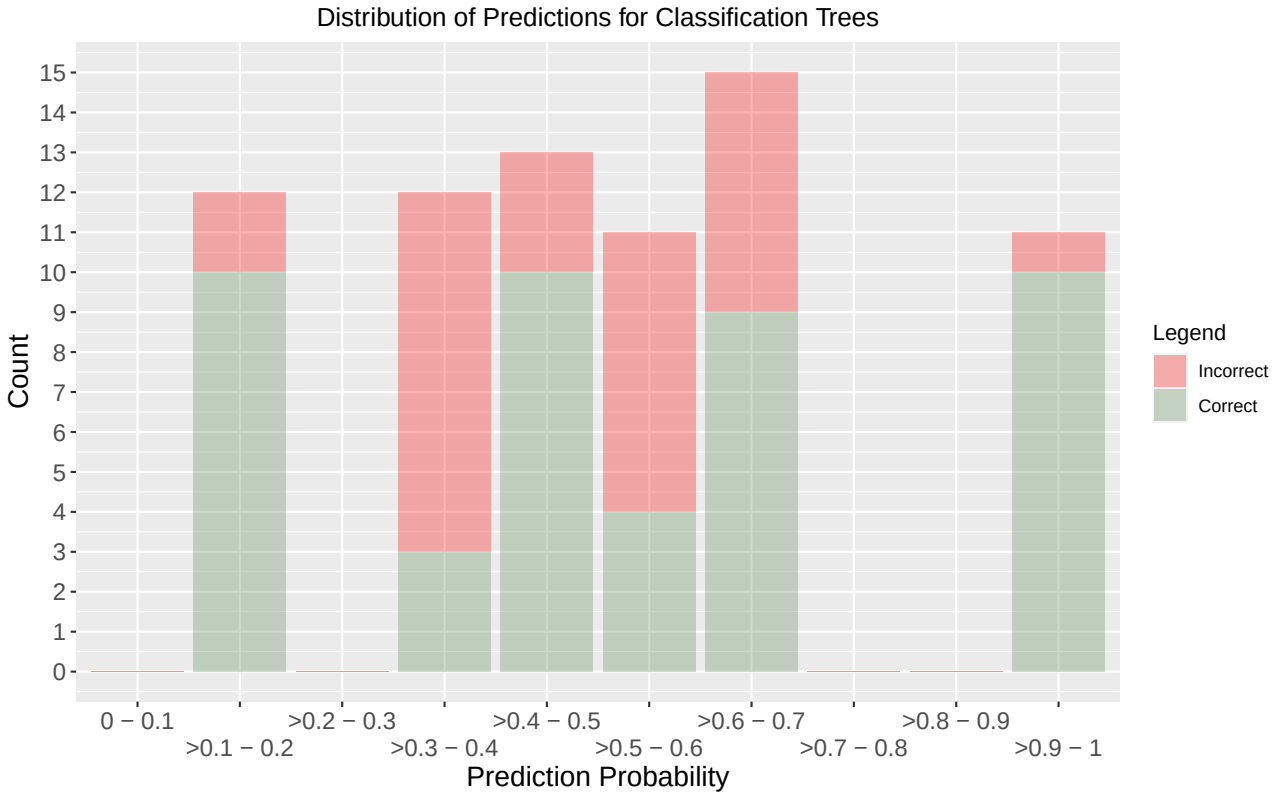


Figure 5.6: Classification tree breakdown of accuracy in different probability ranges

5.3 Bagging/Bootstrap Aggregation

Bagging removes any chance of a visualisation like with classification trees. However, the reduction in understandability is usually made up for with better model performance.

5.3.1 Training of Model

With bagging, there is repeated resampling of the data. Since averaging is then done over many samples, it needs to be shown that the number of samples is enough.

Figure 5.7 shows how the OOB error for the bagging model changes as the number of trees increases. The OOB error seems to become stable from long before the 1000 mark in terms of number of trees.

5.3.2 Final Model

1000 trees will be used for this model. Bagging allows the calculation of importance measures, which, for each predictor, records the amount by which the splitting criterion is improved for each tree, and the average is then taken over all the trees. Figure 5.8 shows a breakdown of the bagging variable importance here.

Here, the first innings is clearly the most important factor. However, most of the other variables do have a part to play. Time does not play a significant part, and interestingly, neither does the toss.

5.3.3 Results

The accuracy of the test set is 64.86%. This shows some improvement to the base classification trees. The Brier score is 0.23.

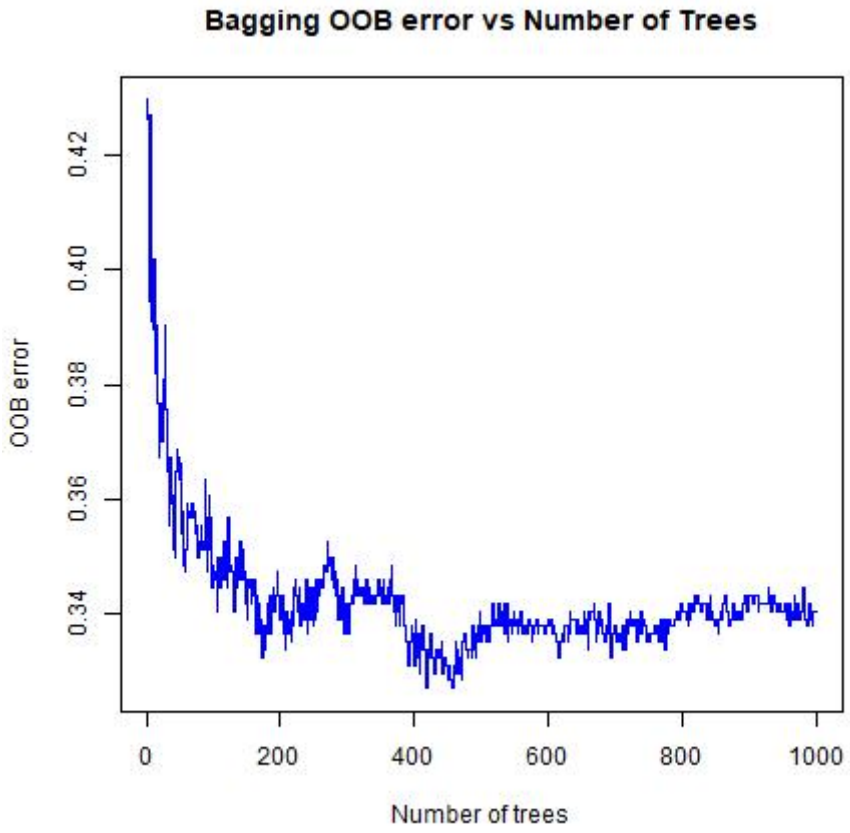


Figure 5.7: Bagging OOB error vs. number of trees

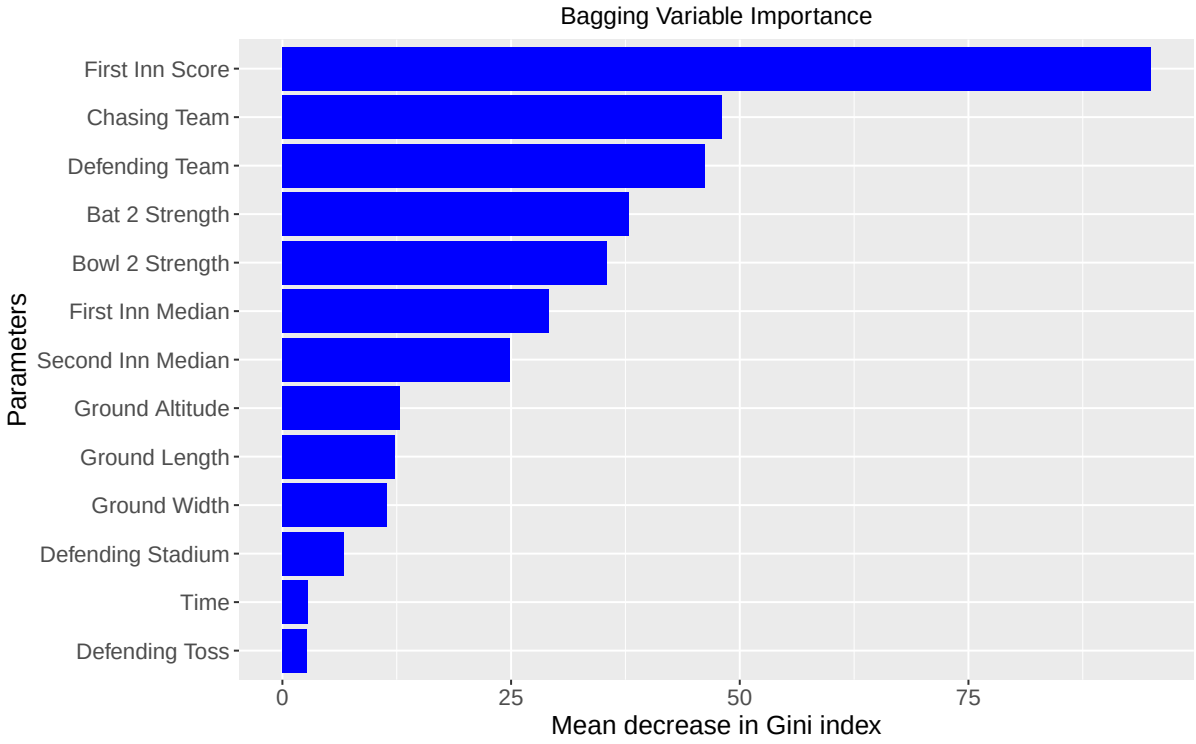


Figure 5.8: Bagging variable importance

Again, the model produces probabilities, so it is possible to do a breakdown of the accuracies. Figure 5.9 shows this breakdown. The model does not seem to perform as well in the extremes as it should. It also seems to make more predictions where it is relatively certain about the

result than the other models. This further supports the view that the model is overfitting - the model is too tailored to the training set and is therefore not generalising well.

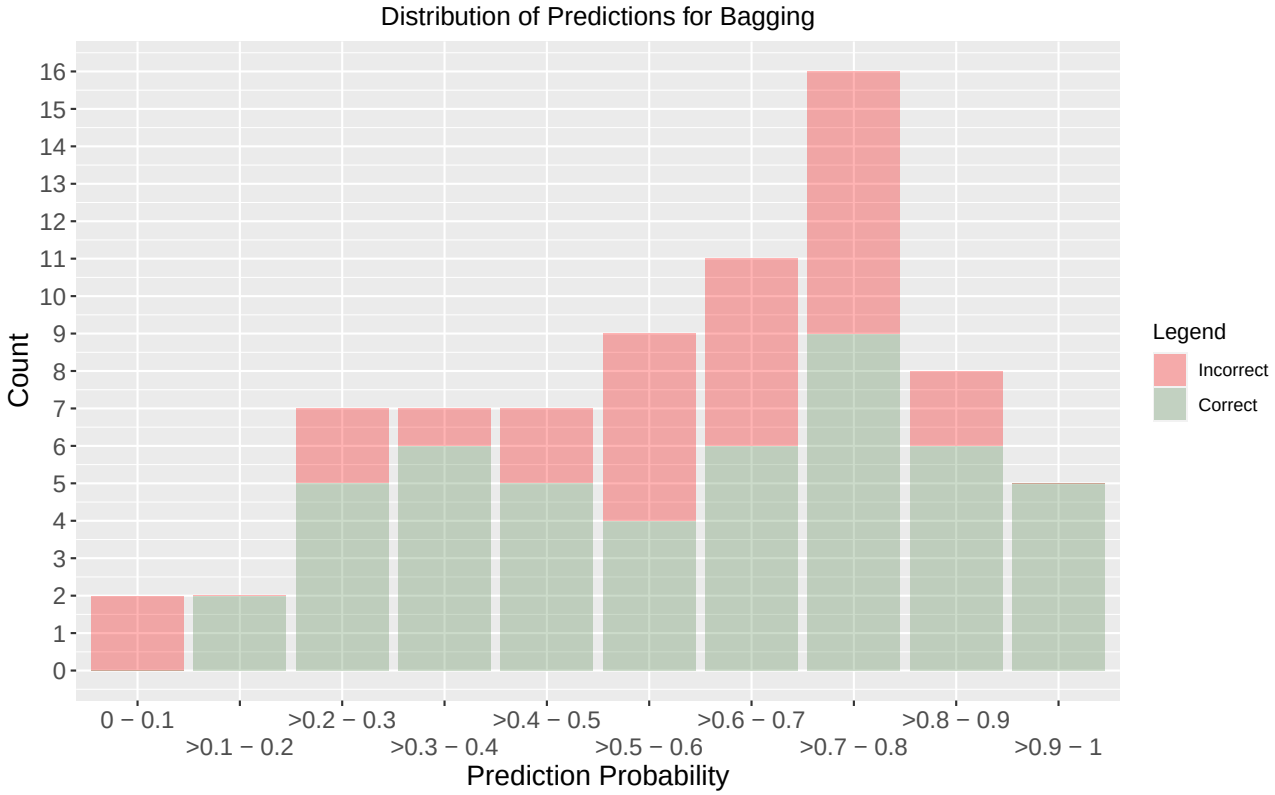


Figure 5.9: Bagging breakdown of accuracy in different probability ranges

As mentioned in the previous section, if there is a very strong predictor, like the first innings score is here, then this predictor will appear in most, if not all, the trees and, therefore, the benefits of the repeated sampling in order to reduce sample variability are removed.

5.4 Random Forest

The next model considered is random forest. By only allowing a random sample, m, of predictors in each tree, random forest removes some of the possible correlation problems with bagging.

5.4.1 Training of Model

Again, it first needs to be shown that the number of trees used is enough, and an optimal value for m needs to be found using cross-validation.

As mentioned in Section 4.5, $m \approx \sqrt{p}$ will be used when testing the number of trees. Figure 5.10 shows how the OOB error changes for the number of trees for random forest. The corresponding bagging results are also added for comparison. As with bagging, the random forest OOB error seems to have become relatively stable long before 1000 trees, so 1000 trees are enough. Random forest is also clearly outperforming bagging. With random forest, there is still the parameter m to be tuned, which can further improve this.

Figure 5.11 looks at how the CV error changes with different values of m. The blue line at $m = 9$ is where the CV error is at its minimum. A value of 9 for m will, therefore, be used.

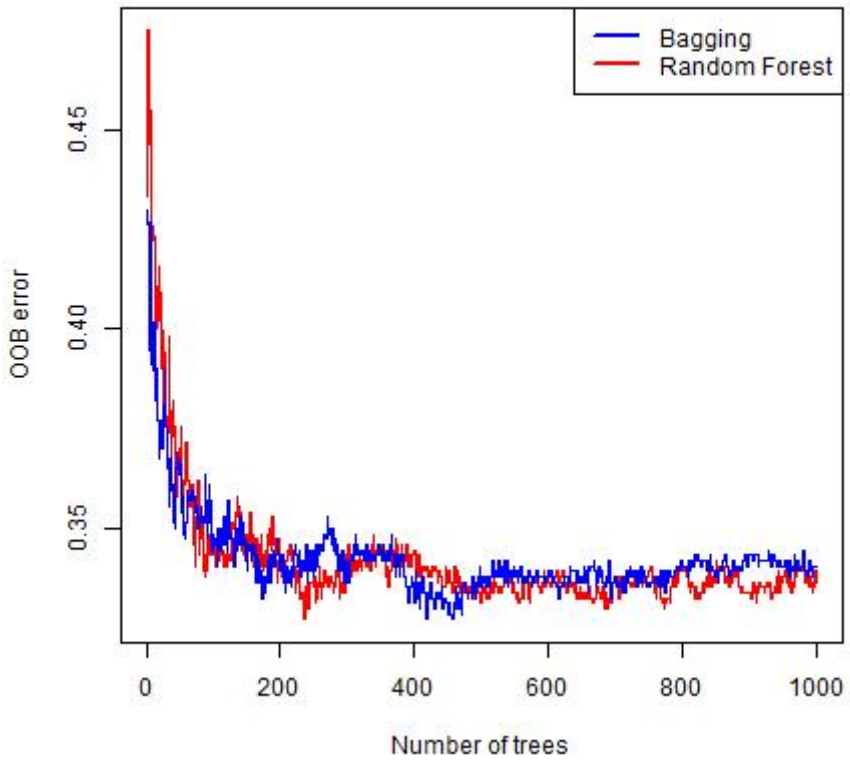


Figure 5.10: Random forest and bagging OOB error vs. number of trees

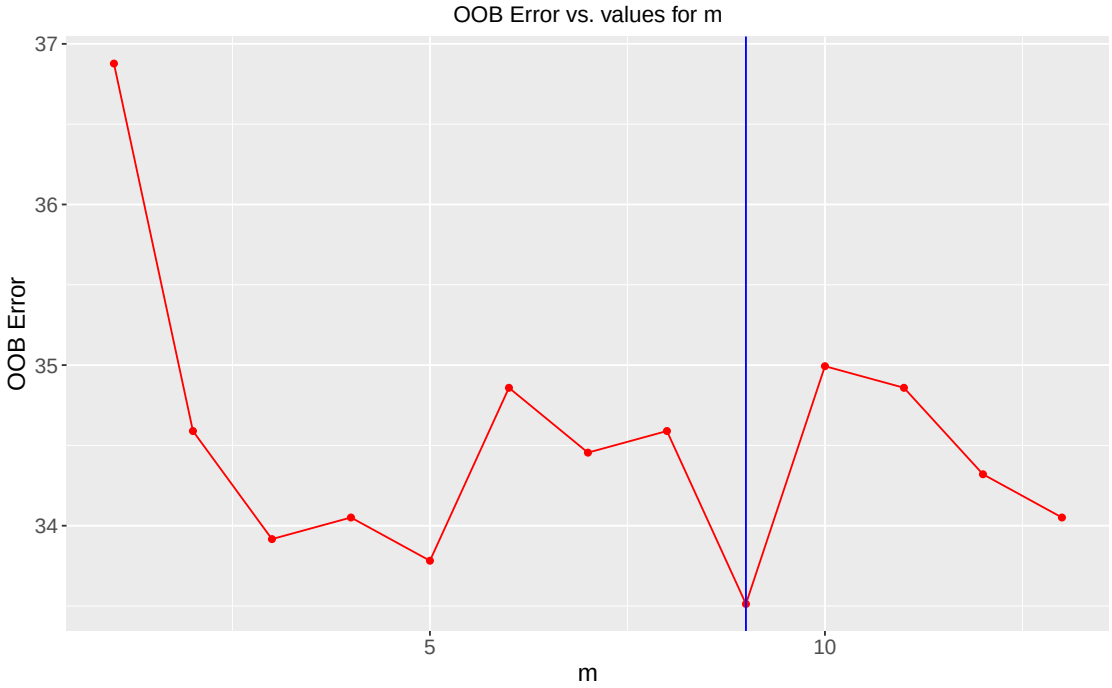


Figure 5.11: Random forest CV error vs. number of terminal nodes

5.4.2 Final Model

The two hyperparameters used are, therefore, $B = 1000$ and $m = 9$.

As with bagging, no single tree can be produced to show how each split is used, but there is

the option of calculating variable importance measures. Figure 5.12 shows the breakdown of variable importance for the random forest model. The results are quite similar to the bagging model. In both of these, the first innings score was by far the most important. Both also have time and defending toss being marked as not very important. All other variables are also almost similar in relative size and order for both models.

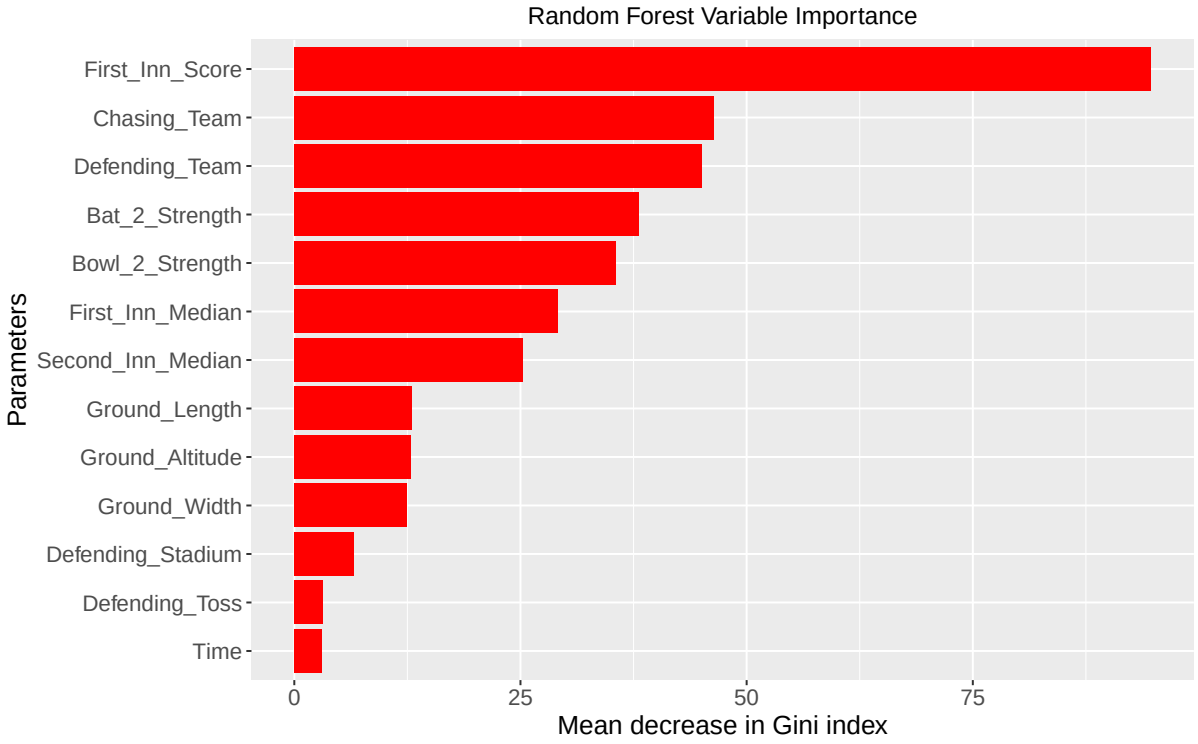


Figure 5.12: Random forest variable importance

5.4.3 Results

The accuracy on the test set for random forest is 67.57%. The Brier score for the model is 0.23.

Figure 5.13 shows a breakdown of the prediction accuracies for random forest. The model performs quite well in most categories. Still, there are some where it performs noticeably poorly, in particular when the model says there is a strong probability (between 0.1 and 0.3) of a defending team loss.

5.5 Boosting

Boosting offers an alternative method of working with trees in which the model learns from the mistakes in previous models. Boosting was done using the extreme gradient boosting implementation.

5.5.1 Training of Model

Boosting was run with a range of parameters, and the best parameters were chosen based on CV error.

The following parameters were tested with the chosen parameters also indicated.

- **Number of trees:** 1000, 1500, 2000, 2500
- **Learning parameter:** 0.05, 0.01, 0.005, 0.001

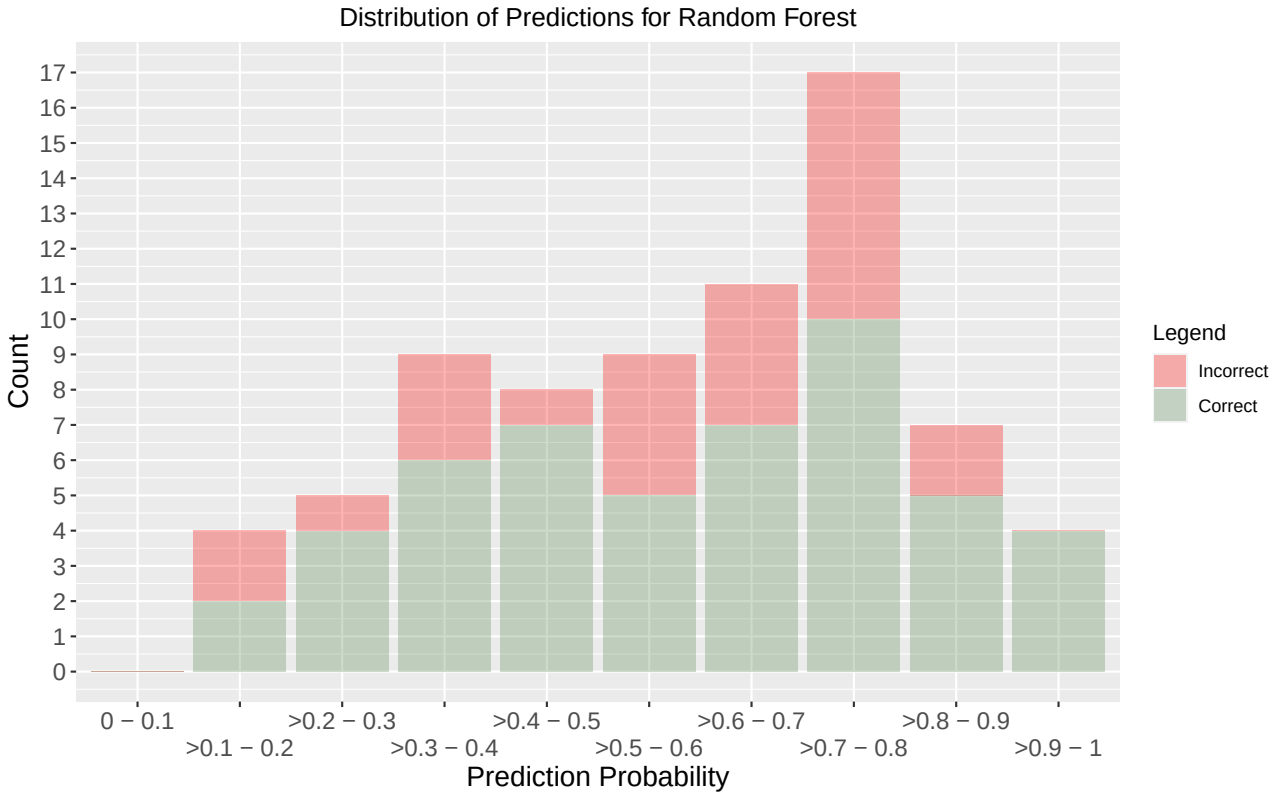


Figure 5.13: Random forest breakdown of accuracy in different probability ranges

- **Interaction depth:** 1 to 6

5.5.2 Final Model

The following parameters were chosen:

- **Number of trees:** 1000
- **Learning parameter:** 0.005
- **Interaction depth:** 2

Figure 5.14 shows the PDPs of features with the largest range on their PDPs. The rest of the PDPs can be found in Appendix A. Again, only the range and shapes of the PDPs are important and not the absolute values. Comparison to the scales of other PDPs again shows that the first innings score is by far the most significant factor. The PDP of the first innings score implies that there are cutoff points below and above which the results are almost certain. There are only a few teams that show any difference when they are defending or chasing in those PDPs. Strength measures move in the expected direction; for example, the stronger the defending team's bowling strength, the more likely the defending team is to avoid defeat.

5.5.3 Results

The accuracy on the test set is 68.92%. The Brier score is 0.2. Figure 5.15 contains a breakdown of the boosting accuracies. The model performs quite well in all ranges, with only low accuracies coming in ranges where the model predicted an uncertain outcome. Even though its performance in these ranges is in line with expected, there are far too many predictions in these ranges, especially compared to logistic regression, which is why the model performs worse than logistic regression.

This marks the end of the tree-based models. Next are support vector machines, artificial neural networks and finally, naive Bayes.

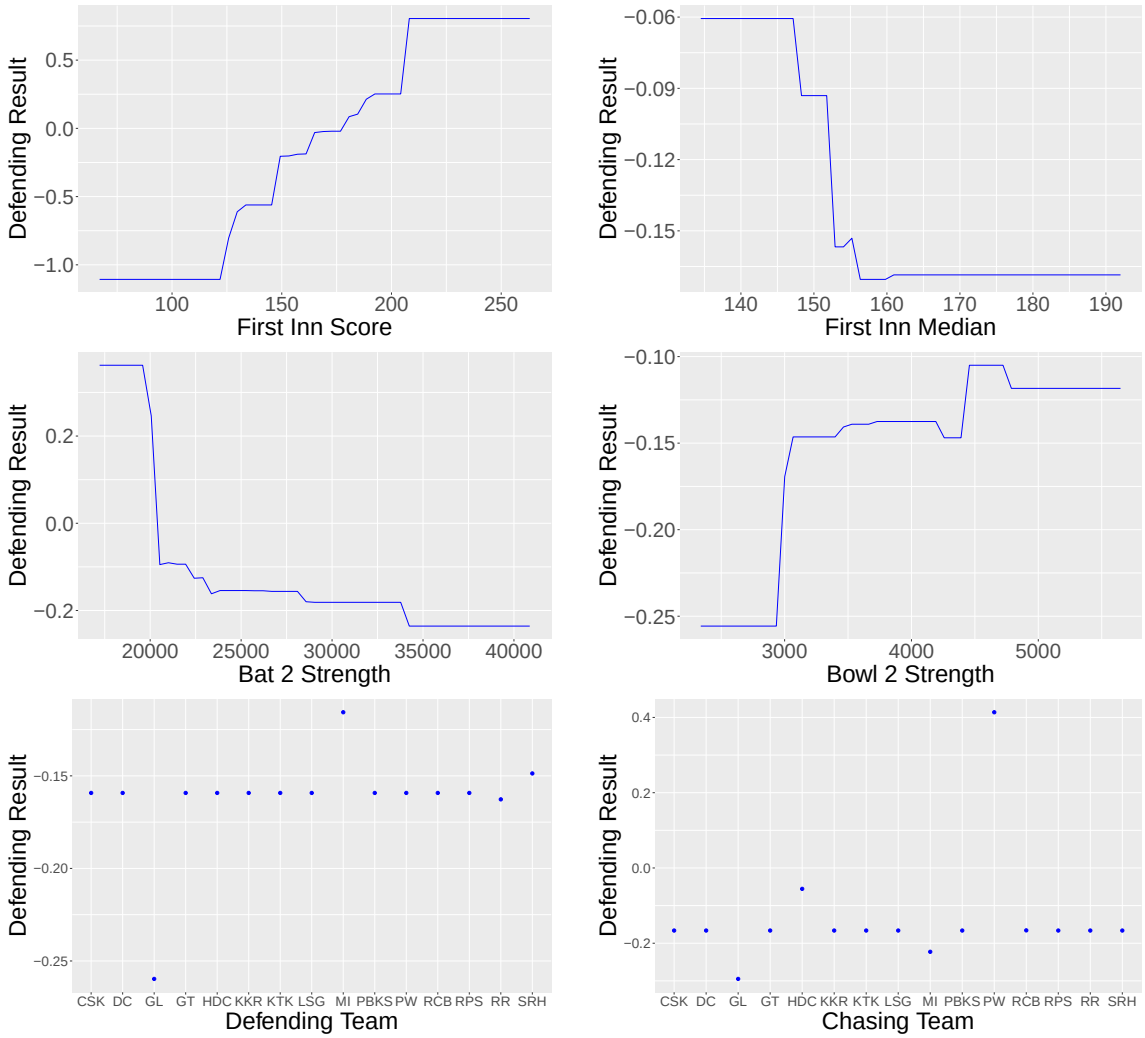


Figure 5.14: Boosting partial dependency plots part 1

5.6 Support Vector Machines

5.6.1 Training of Model

The three hyperparameters that will be tweaked are cost, gamma and kernel. A grid search is used to test each combination of hyperparameters. The values that were tested for each of these parameters are as follows:

- **Cost:** 0.1 to 100 in steps of 0.1
- **Gamma:** 0.001, 0.01, 0.1, 1, 10
- **Kernel:** Linear, Radial, Sigmoid

5.6.2 Final Model

The following parameters were chosen based on cross-validation:

- **Cost:** 0.8
- **Gamma:** 0.001
- **Kernel:** Linear

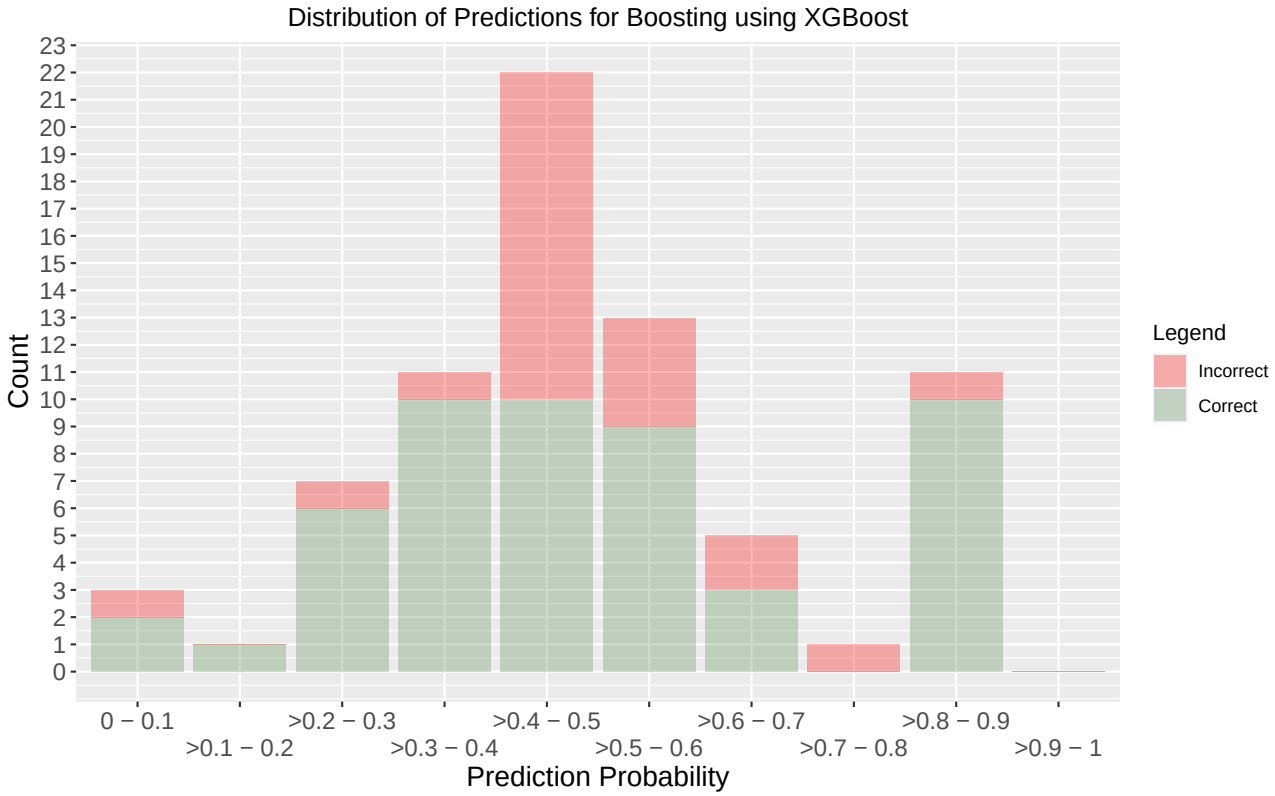


Figure 5.15: Boosting using extreme gradient boosting breakdown of accuracy in different probability ranges

5.6.3 Results

The accuracy on the test set is 68.92%. With support vector machines, predictions do not come with probabilities. This means that the Brier score cannot be calculated and that there can be no breakdown of accuracies of predictions in the different probability groups. Nevertheless, there are models that performed better on the test set, so there is no need to explore this model further.

5.7 Neural Networks

The next model considered is neural networks.

5.7.1 Training of Model

With neural networks, there are endless possibilities for the architecture itself in terms of number of hidden layers and number of nodes in each layer. There are also various hyperparameters that need to be chosen, such as activation functions, learning rate, early stopping, drop-out and weight decay. There were also many parameters that needed to be calculated, particularly due to the large number of categorical variables, which all required one hot encoding. The main contributor to the large number of parameters was the team names, which were needed for both the chasing team and defending team categories.

It is, therefore, impractical to test every possible combination of hyperparameters and architectures. Instead, a trial and error approach was taken to get a feel of how different hyperparameters affected the model.

The following were findings when dealing with the possible models:

- If there was more than one hidden layer, the model failed to converge even with 100,000 steps.
- No hidden layer performed better than one hidden layer.
- Changes to other hyperparameters did not improve the model.

The model may not convergence with additional hidden layers due to time and computer power restraints but could also be due to there not being enough observations to learn more complex patterns in the data.

5.7.2 Final Model

After much testing using trial and error, the following model was chosen:

- No hidden layers.
- Learning rate of 1.
- No changes to batch size.
- No early stopping, drop-out or weight decay.

5.7.3 Results

The accuracy on the test set was 67.57%. The Brier score was 0.21. The breakdown of prediction accuracies can be seen in Figure 5.16.

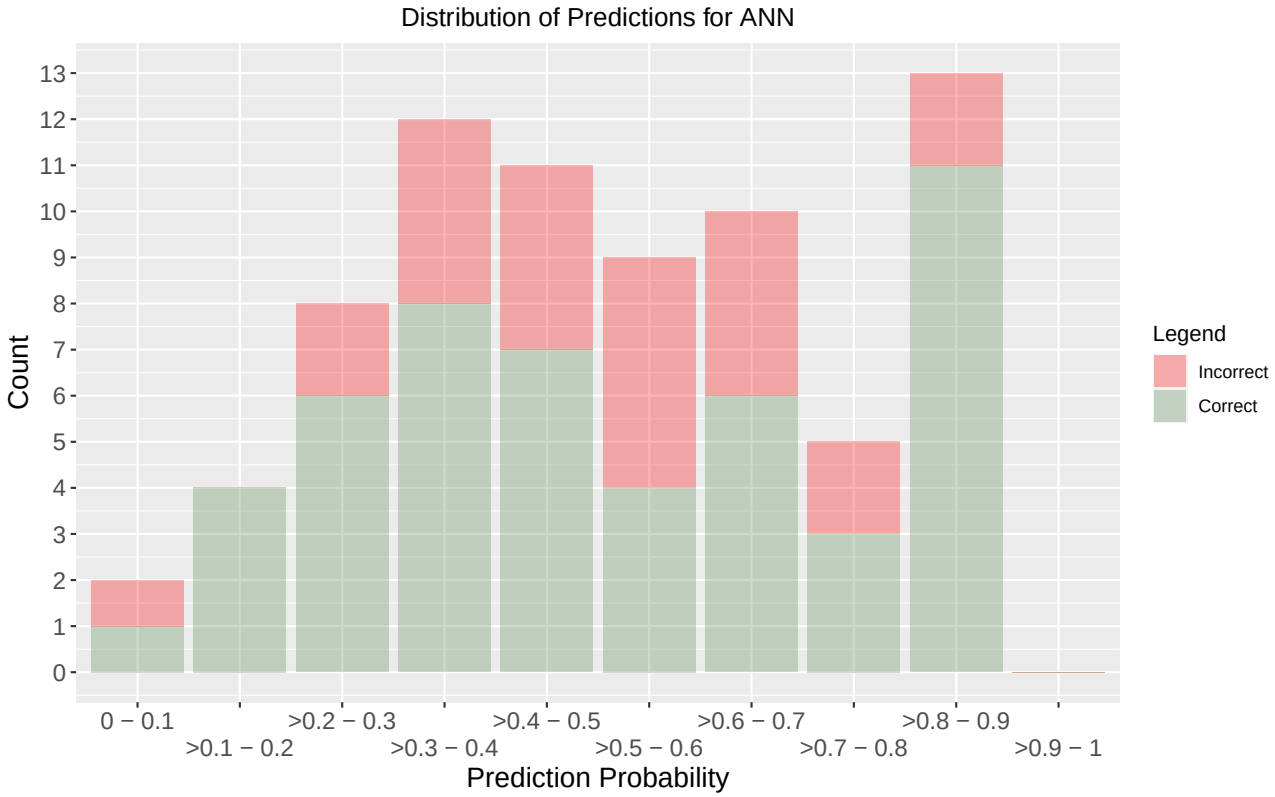


Figure 5.16: Artificial neural network breakdown of accuracy in different probability ranges

Neural networks performed quite well across all groups, especially when it was quite confident of a result. For those that it was not confident one way or another, the results reflected that. However, the model performed slightly worse than the more robust models in ranges where it thought the results could go either way.

5.8 Naive Bayes

Naive Bayes is the final model that is considered.

5.8.1 Training of Model

With naive Bayes, there are no hyperparameters that need to be tested.

5.8.2 Final Model

The only hyperparameter that had to be set was the smoothing constant, α, which was set at 1.

5.8.3 Results

The accuracy on the test set is 56.8%. The Brier score is 0.32. The relatively poor performance of the model compared to other models could be due to the assumption that the parameters are independent. As mentioned in the previous section, there are a couple of pairs of variables that are highly correlated. In addition, many variables did have some correlation, even if it was not high. The Brier score here is even lower than the random predictor score of 0.25.

The breakdown of performance in the different prediction probability ranges can be seen in Figure 5.17. The model incorrectly predicts more wins for the defending team than other models. The model also makes many strong predictions for the defending team, which does not end up playing out.

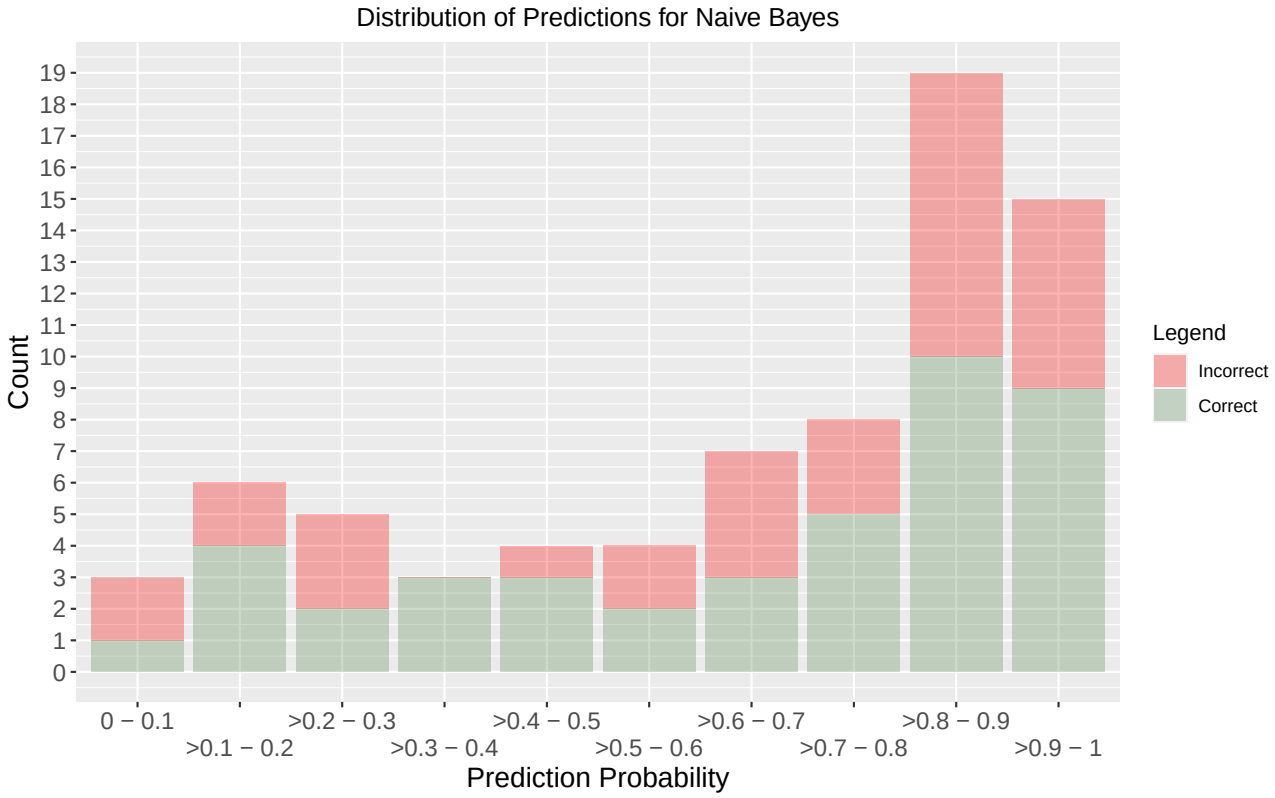


Figure 5.17: Naive Bayes breakdown of accuracy in different probability ranges

5.9 Results Comparison

Table 5.2 contains a breakdown of the performance of the various models. Only the test accuracy and Brier score are used in the selection of models. Logistic regression performs the best in terms of accuracy and on the Brier score. The Brier score listed is rounded to two decimal places. Both logistic regression and boosting have a value of 0.2 but it is worth noting that to three decimal places the Brier score for logistic regression is 0.196 and for boosting it is 0.2. So logistic regression does indeed perform the best in both the test accuracy and Brier score.

Model	Test Accuracy	Brier Score
Logistic Regression	70.27%	0.2
Classification Trees	62.16%	0.22
Bagging	64.86%	0.23
Random Forest	67.57%	0.23
Boosting	68.92%	0.2
Support Vector Machines	68.92%	n/a
Neural Networks	67.57%	0.21
Naive Bayes	56.76%	0.32

Table 5.2: Model performance comparison

Looking at the breakdown of the accuracies chart for logistic regression shows that the model only really has somewhat poor performance when the model itself says that there is not a clear winner. Even then, it performs in line with its expected accuracy at those ranges. The model, therefore, seems capable of performing the initial aim of this paper, which was to build a model that allowed teams to see what scores would be needed for different confidence levels to get a particular result.

Logistic regression, being the top performing model, is somewhat surprising given that it assumes a linear relationship between the independent and dependent variables. There were also no interaction terms used between features in this paper.

The worst performing model is naive Bayes, which seems to imply that the assumption of independence was too strong.

5.10 Model Application

Besides giving a probability of the defending team avoiding a loss after the first innings the model can also be used by teams before that point to assess what runs to target to attain a certain confidence of avoiding a loss in regular time. At the start of every match, the various relevant factors available before the match can be input into the model, and the confidence of avoiding defeat in regular time for the team batting first at different totals can be computed. For example, consider the first match in the test set where Chennai Super Kings batted first against Kolkata Knight Riders. After inputting all factors available at the start of the match they would be able to generate a table with the confidence of avoiding defeat in regular time for various first innings totals. Table 5.2 contains an abbreviated version of this with a few confidence levels.

In the match, Chennai Super Kings only scored 131 which according to the model only gave them a 0.29 probability of avoiding defeat in regular time. They did indeed lose as Kolkata Knight Riders won by 6 wickets with 9 balls remaining.

Confidence	Total
20	117
40	145
60	168
80	195

Table 5.3: Confidence in avoiding loss for CSK batting first against KKR in the first match of the 2022 IPL season

5.11 Conclusion

The aim of this chapter was to apply various machine learning methods to predict the results of IPL matches at the halfway stage and see what value could be derived in terms of confidence in these predictions. Various models were explored, and logistic regression performed the best on the test data set, with naive Bayes performing the worst. In the final chapter all the work performed in this study is reviewed, possible future work is detailed and finally concluding remarks are provided.

Chapter 6

Conclusion

In this final chapter, the work done in this study is reviewed. Possible future steps are then explored by ways to expand the study or things that could have been done differently. Finally, concluding remarks are provided by looking at whether the specific aims of the paper were met and the implications of this paper.

6.1 Review of Study

This paper set out to answer the question of how many runs are enough for a team batting first in the IPL. The aim was to build a model that would determine how many runs would be needed for a certain confidence of avoiding defeat.

For this study, all IPL matches played from the 2008 season until the 2022 season were used with the 2022 season used as the test set. Data was gathered by web-scraping the scorecards and extracting the information needed to generate key features. Some of the other required information was gathered from various online sources. The code used to extract the information was kept as flexible as possible to allow the testing of models on various other tournaments and formats in the future. Custom measures of batting and bowling strengths were also formulated in order to capture the strengths of the players involved.

Various machine learning methods were explored with various hyperparameters tested. The machine learning models used were: logistic regression, classification trees, bagging, random forest, boosting, support vector machines (SVMs), artificial neural networks (ANNs) and naive Bayes.

Logistic regression came out as the top performing model able to generate a prediction accuracy of 70.27% on the 2022 IPL season which came with a Brier score of 0.2. Most of the incorrect predictions occurred when the model itself predicted a probability that implied that both outcomes were equally likely and the number of incorrect results was in line, more or less, with the predicted probabilities. It was also demonstrated how the model could be used to determine the totals needed for the team batting first for different confidence levels of avoiding defeat in regular time.

6.2 Possible Future Steps

While the model was built on the IPL dataset because of the availability of data, there is no reason for the model not to work on other T20 and even one-day data sets.

There were various factors mentioned in Section 2.3 that would have been used if available but were not possible for reasons discussed in that section. These factors included pitch conditions, temperature, humidity, precipitation and sentiment analysis.

Logistic regression emerged as the top performing model without incorporating interaction terms. Various interaction terms between variables could be tested to see if there is any improvement to the model. For example, as mentioned in Section 2.3 Bandulasiri (2008) noted that the toss was only significant for day/night matches.

In Section 3.4.2 it was mentioned there could be a minimum number of balls where a player's strength is only used after that point. However, as mentioned, that would require some extra thought into how that would fit in with the current scaling methods. As an alternative scaling-in could be done with the number of balls bowled rather than the number of matches played. A player then would only have their full strength used after a certain number of balls. These alternatives would better deal with the outlier case of Sachin Baby.

As mentioned in Section 5.7, the training of the ANNs was impacted by the power of the machines used and the time available. Perhaps with more time and resources, other neural network structures and hyperparameters could be tested, and some of these might lead to better performance. Some models such as ANNs require large amounts of data and could perform much better as there are more seasons to train on.

In Section 3.2.1 it was pointed out that there was scope to adjust the parameters for how much to weight the form of players in player strength calculations (μ) and how many games to calculate the form of players over (n). n was also used for how many matches were used to scale in the median scores for stadiums and the player strengths for new players. These parameters could be tweaked using a grid search, although each new combination would require rerunning all models, so it would again require lots of time and computing power.

Different formulae could be used to capture the batting and bowling proportion of players. One possible method for the batting proportion would be to compare the number of balls faced by a player to the total number of balls faced by the team. The batting proportion would then be the square root of the proportion of balls faced by that player. Similarly, the bowling proportion can be defined in terms of how many balls a bowler has bowled compared to the total number of balls bowled by the team. The bowling proportion would then be the square root of the proportion of balls bowled by that player. This version of the bowling proportion would be much simpler to understand and implement, and together, the batting and bowling proportions would be consistent.

It would also be interesting to compare the probability predictions after the first innings, the betting odds at that stage and the eventual outcomes to determine whether such a model could give one an advantage in the betting market.

6.3 Concluding Remarks

The research objective stated in Section 1.6 was to develop a model that can provide first innings total targets for different confidence levels of avoiding defeat in regular time. The research objective has therefore been achieved with the logistic regression model developed. The sub-objectives have also been met:

1. The current research around sports predictions has been understood.
2. The data has been understood.
3. Machine learning models for modelling of first innings totals in the IPL have been developed.

The logistic model developed has a real-world use in that IPL teams can use this to set final targets at various stages of the first innings in order to obtain a certain confidence in avoiding defeat in regular time. At the start of every match, the various relevant factors

available before the match can be input into the model, and the confidence of avoiding defeat in regular time for the team batting first at different totals can be computed. This information can be used by both the team batting first and the team bowling first. This will help the team batting first pace their innings and not take unnecessary risks. Similarly, the team bowling first can decide how attacking or defensive they should be at various points in the innings. The bowling team may choose to alter the model slightly to be stated in terms of the chasing team avoiding loss in regular time. The model could also be used by those in the betting market but further study is needed against the actual betting odds available after the first innings of matches.

Appendix A

Boosting Partial Dependency Plots

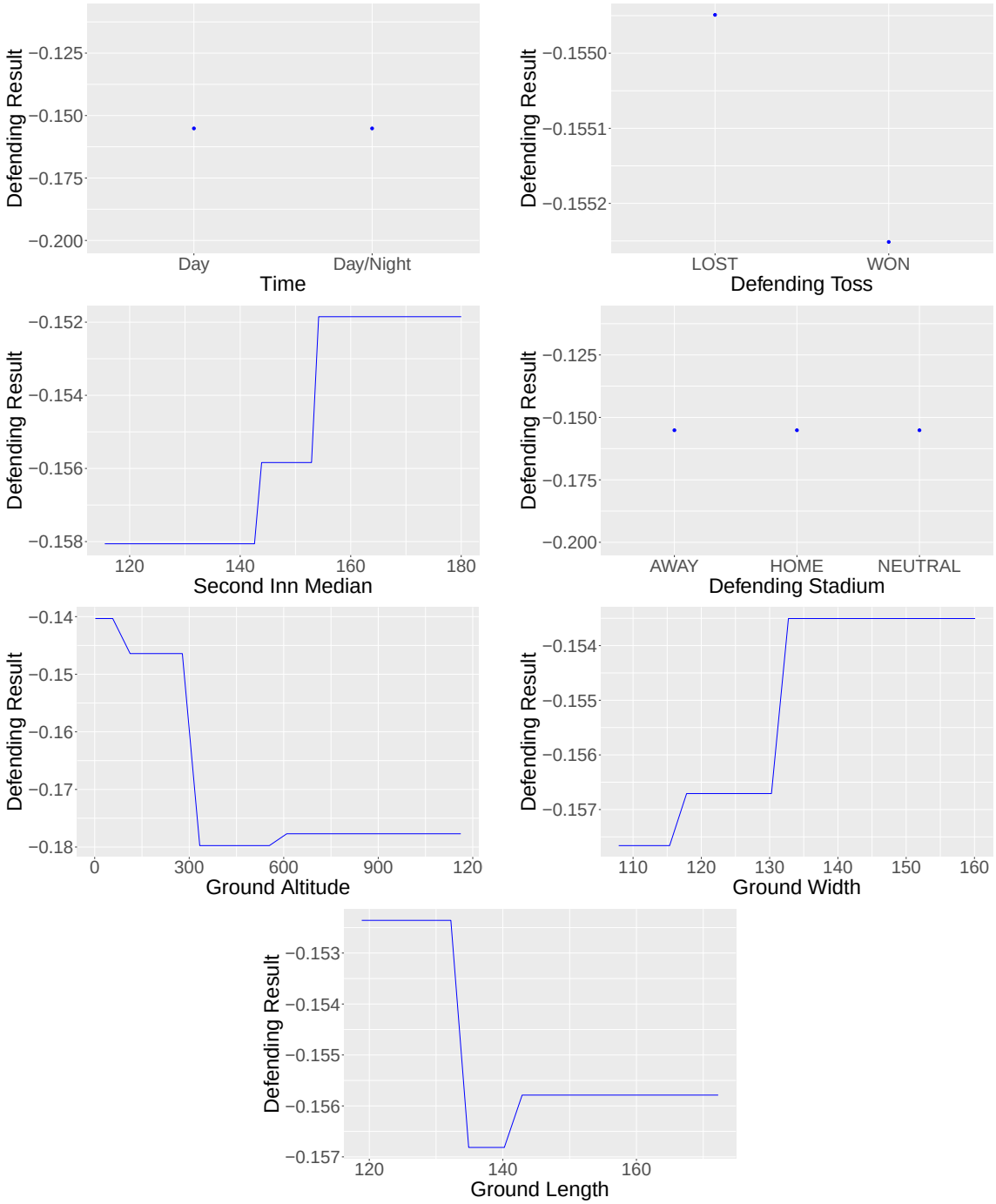


Figure A.1: Boosting partial dependency plots part 2

Bibliography

Alpaydin, Ethem (2020). *Introduction to Machine Learning*. MIT press.

Bandulasiri, Ananda (2008). "Predicting the Winner in One Day International Cricket". In: *Journal of Mathematical Sciences & Mathematics Education* 3.1, pp. 6–17.

Berrar, Daniel (Jan. 2018). "Cross-Validation". In: ISBN: 9780128096338. DOI: 10.1016/B978-0-12-809633-8.20349-X.

Bishop, Christopher M and Nasser M Nasrabadi (2006). *Pattern Recognition and Machine Learning*. Vol. 4. 4. Springer.

Breiman, Leo (2001). "Random forests". In: *Machine Learning* 45, pp. 5–32.

Brier, Glenn W et al. (1950). "Verification of forecasts expressed in terms of probability". In: *Monthly Weather Review* 78.1, pp. 1–3.

Bunker, Rory P. and Fadi Thabtah (2019). "A machine learning framework for sport result prediction". In: *Applied Computing and Informatics* 15.1, pp. 27–33. ISSN: 2210-8327. DOI: https://doi.org/10.1016/j.aci.2017.09.005. URL: https://www.sciencedirect.com/science/article/pii/S2210832717301485.

Chen, Tianqi and Carlos Guestrin (2016). "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Chen, Tianqi et al. (2015). "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* 1.4, pp. 1–4.

Deshpande, Nilkanth, Shilpa Gite, and Rajanikanth Aluvalu (Apr. 2021). "A review of microscopic analysis of blood cells for disease detection with AI perspective". In: *PeerJ Computer Science* 7, e460. DOI: 10.7717/peerj-cs.460.

Duckworth, Frank C and Anthony J Lewis (1998). "A fair method for resetting the target in interrupted one-day cricket matches". In: *Journal of the Operational Research Society* 49.3, pp. 220–227.

El Naqa, Issam and Martin J. Murphy (2015). "What Is Machine Learning?" In: *Machine Learning in Radiation Oncology: Theory and Applications*. Ed. by Issam El Naqa, Ruijiang Li, and Martin J. Murphy. Cham: Springer International Publishing, pp. 3–11. ISBN: 978-

3-319-18305-3. DOI: 10.1007/978-3-319-18305-3_1. URL: https://doi.org/10.1007/978-3-319-18305-3_1.

Er, Şebnem (2020). *Lecture 7: Support Vector Machines.*

First Post (2022). *IPL 2022: Gujarat Titans bag Rs 20 crore winner's cheque; full list of award winners and prize money details.* URL: https://www.firstpost.com/firstcricket/sports-news/ipl-2022-full-list-of-award-winners-prize-money-details-and-important-stats-10736281.html#:~:text=Debutant%20franchise%20Gujarat%20Titans%20scripted,the%20post%2Dfinal%20award%20ceremony. (visited on 05/30/2022).

Gençay, Ramazan and Min Qi (Aug. 2001). "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging". In: *Neural Networks, IEEE Transactions on* 12, pp. 726 –734. DOI: 10.1109/72.935086.

Gollapudi, Nagraj (2022). *IPL unveils new format for 2022, with two groups and seedings.* URL: https://www.espncricinfo.com/story/ipl-2022-unveils-new-format-with-two-groups-and-seedings-1302674 (visited on 02/25/2022).

Gómez-Ruano, M.A., R. Pollard, and C. Lago-Peñas (2021). *Home Advantage in Sport: Causes and the Effect on Performance.* Taylor & Francis. ISBN: 9781000462050. URL: https://books.google.co.za/books?id=hQxIEAAAQBAJ.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning.* MIT press.

Hachimi, Marouane et al. (Apr. 2020). "Multi-stage Jamming Attacks Detection using Deep Learning Combined with Kernelized Support Vector Machine in 5G Cloud Radio Access Networks". In.

Ho, Tin Kam (1995). "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition.* Vol. 1. IEEE, pp. 278–282.

Holliday, Jeffrey J (2012). *The emergence of L2 phonological contrast in perception: The case of Korean sibilant fricatives.* The Ohio State University.

Horvat, Tomislav and Josip Job (Sept. 2020). "The use of machine learning in sport outcome prediction: A review". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10, e1380. DOI: 10.1002/widm.1380.

Horvát, Emőke-Ágnes (Jan. 2013). "Modelling and inferring connections in complex networks". In.

James, Gareth et al. (2013). *An introduction to statistical learning.* Vol. 112. Springer.

Jeong, Dong-Hwa et al. (July 2022). "A Comparative Study on the Influence of Undersampling and Oversampling Techniques for the Classification of Physical Activities Using an Imbalanced Accelerometer Dataset". In: *Healthcare* 10, p. 1255. DOI: 10.3390/healthcare10071255.

Jhawar, Madan and Vikram Pudi (Aug. 2016). "Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach". In: Machine Learning and Data Mining for Sports Analytics, ECML-PKDD'16.

Johnson, N. et al. (May 2020). "Machine Learning for Materials Developments in Metals Additive Manufacturing". In.

Kaluarachchi, Amal and Aparna Varde (Dec. 2010). "CricAI: A classification based tool to predict the outcome in ODI cricket". In: pp. 250–255. ISBN: 978-1-4244-8549-9. DOI: 10.1109/ICIAFS.2010.5715668.

Kampakis, Stylianos and William Thomas (2015). *Using machine learning to predict the outcome of english county twenty over cricket matches*. arXiv: 1511.05837 [stat.ML].

Kapadia, Kumash et al. (Nov. 2019). "Sport Analytics for Cricket Game Results using Machine Learning: An Experimental Study". In: *Applied Computing and Informatics* ahead-of-print. DOI: 10.1016/j.aci.2019.11.006.

Khare, Shubham (2020). *IPL Playoffs Format: Explaining the concept*. URL: https://www.sportskeeda.com/cricket/ipl-playoffs-format-explaining-the-concept (visited on 10/27/2020).

Krogh, Anders and John Hertz (1991). "A simple weight decay can improve generalization". In: *Advances in neural information processing systems* 4.

Kumar, Rajesh, Aftab Ul Nabi Shahani, and Karchi (Jan. 2020). "Bank Credit Risk Management Using Machine Learning Algorithms". In.

Kumar, Saurabh et al. (2021). "Cricket Data Analytics and Prediction". In.

Kumar, Sourav (2014). *A batting index for limited-overs cricket*. URL: https://www.espncricinfo.com/story/rankings-a-batting-index-for-limited-overs-cricket-752205 (visited on 06/13/2014).

Laghate, Gaurav (2020). *Almost half of Indian TV viewers watched IPL 2020*. URL: https://economictimes.indiatimes.com/industry/media/entertainment/almost-half-of-indian-tv-viewers-watched-ipl-2020/articleshow/79323814.cms?from=mdr (visited on 11/21/2020).

Lewis, Michael (2004). *Moneyball: The art of winning an unfair game*. WW Norton & Company.

Luhaniwal, Vikashraj (2019). *Analyzing different types of activation functions in neural networks — which one to prefer?* URL: https://towardsdatascience.com/analyzing-different-types-of-activation-functions-in-neural-networks-which-one-to-prefer-e11649256209 (visited on 02/13/2023).

Messenger, Robert and Lewis Mandell (1972). "A modal search technique for predictive nominal scale multivariate analysis". In: *Journal of the American statistical association* 67.340, pp. 768–772.

Morgan, James N and John A Sonquist (1963). "Some results from a non-symmetrical branching process that looks for interaction effects". In: *Young* 8.5.

Mustafa, Raza Ul et al. (2017). "Predicting The Cricket Match Outcome Using Crowd Opinions On Social Networks: A Comparative Study Of Machine Learning Methods". In: *Malaysian Journal of Computer Science* 30, pp. 63–76.

Natrella, Mary et al. (2010). "NIST/SEMATECH e-handbook of statistical methods". In: *Nist/Sematech* 49.

New Zealand Cricket (2014). *What's WASP all about?* URL: https://www.nzc.nz/news-items/archive/whats-wasp-all-about (visited on 03/20/2021).

Nimmagadda et al. (2018). "Cricket score and winning prediction using data mining". In: *International Journal of Advance Research and Development* 3.

Reider, Bruce (2014). "Moneyball". In: *The American Journal of Sports Medicine* 42.3. PMID: 24585673, pp. 533–535. DOI: 10.1177/0363546514524161. eprint: https://doi.org/10.1177/0363546514524161. URL: https://doi.org/10.1177/0363546514524161.

Ren, Qiubing, Mingchao Li, and Shuai Han (Feb. 2019). "Tectonic discrimination of olivine in basalt using data mining techniques based on major elements: a comparative study from multiple perspectives". In: *Big Earth Data* 3, pp. 1–18. DOI: 10.1080/20964471.2019.1572452.

Seltman, Howard J (2012). *Experimental design and analysis*.

Silva, Basil and Tim Swartz (Mar. 1998). "Winning the Coin Toss and the Home Team Advantage in One-Day International Cricket Matches". In: *The New Zealand Statistician* 32.

Singhal, Gaurav (2020). *Ensemble Methods in Machine Learning: Bagging Versus Boosting*. URL: https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting (visited on 10/27/2020).

Smith, Matthew (2021). *The Exponenetial Rise of T20 cricket*. URL: https://www.sportingferret.com/2021/04/15/the-exponential-rise-of-t20-cricket/ (visited on 04/15/2021).

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Stern, Steven (Apr. 2016). "The Duckworth-Lewis-Stern method: extending the Duckworth-Lewis methodology to deal with modern scoring rates". In: *Journal of the Operational Research Society* 67. DOI: 10.1057/jors.2016.30.

Tekade, Pallavi et al. (2020). "Cricket match outcome prediction using machine learning". In: *International Journal* 5.7.

The Hindu (2019). *Why the Indian Premier League has been such a hit*. URL: https://www.thehindu.com/thread/sports/why-the-indian-premier-league-has-been-such-a-hit/article26920209.ece (visited on 04/23/2019).

Viswanadha, Sasank et al. (2017). “Dynamic Winner Prediction in Twenty20 Cricket: Based on Relative Team Strengths.” In: *MLSA@ PKDD/ECML*, pp. 41–50.

Zheng, Alice and Amanda Casari (2018). *Feature engineering for machine learning: principles and techniques for data scientists.* ” O’Reilly Media, Inc.”