University of Cape Town

Engineering Features for Tennis Match Outcome Prediction in a Statistical Learning Framework

Students:
Adam Bresler
Luka Elliott

Supervisor:
Stefan Britz

A project submitted in fulfillment of the requirements for the degree of Honours in Statistics
in the

Department of Statistical Sciences

December 2, 2020

Declaration of Authorship

We, Adam Bresler and Luka Elliott, declare that this thesis titled, "Engineering Features for Tennis Match Outcome Prediction in a Statistical Learning Framework" and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for an honours degree at the University of Cape Town.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.

Name: Adam Bresler

Signed:

Date: 2 December 2020

Name: Luka Elliott

Signed:

Date: 2 December 2020

Abstract

As the sports betting industry booms, predicting the outcome of tennis matches is becoming increasingly lucrative. There has been extensive research into the prediction of tennis match outcomes. However, most of the current literature investigates which supervised learning technique makes the most accurate predictions using raw historical data. While feature engineering is often used, the focus tends to be placed on predictive models. This project places greater emphasis on engineering predictive features that capture the fundamental dynamic of a tennis match, specifically male singles tennis matches on the ATP tour. We manipulate the raw data in order to consider many features, such as the serve and return match play of tennis, critical moments such as break points, comparing the players within a match, as well as adjusting for factors such as match length. Using the resulting dataset, we deploy various methods of creating an historical record. These include introducing a recency bias and stratifying by court in order to predict tennis matches using historical data.

From this data, four supervised learning models are used to create match outcome predictions. Using a test set of matches played in 2019, we were able to correctly classify 63.89% of tennis matches. The engineered features did slightly improve the classification rate compared to the raw features, albeit not by a significant margin. Furthermore, there was a minimal difference between the supervised learning models. While our engineered features did not make major improvements to prediction accuracy compared to the raw features, we believe there is scope for improvement in the prediction of tennis match outcomes. This improvement does not need to stem from advanced models, but can be found by engineering features that better capture the dynamics of a tennis match.

Acknowledgements

We'd like to thank our supervisor Mr Stefan Britz for his assistance throughout this project. His shared passion for tennis made this experience all the more enjoyable, enabling us to learn more than we had imagined.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Tennis is one of the most popular sports in the world with a global following of an estimated 1 billion fans (World Atlas, 2020). Each year the Association of Tennis Professionals (ATP) hosts around 64 tournaments (ATP Tour, 2020), where fans are treated to watching the likes of Federer, Nadal and Djokovic. Due to the major following of tennis, betting on tennis match outcome has become a major industry. In 2012, £58 million worth of bets were placed on the Wimbledon Men's Singles Final alone (Glass, 2012). Thus, predicting the outcome of sporting events using analytical techniques is incredibly lucrative, with sports betting currently making up approximately 70% of global gambling revenue (Yogonet, 2020).

Before delving into any form of statistical analysis, it is important to consider the game of tennis purely from a sporting standpoint. It is vital to have a grasp of the nuances of tennis such as the rules, the scoring system and tennis jargon, as the relevant terminology will be used throughout this project. After gaining a familiarity with the sport of tennis, we will then delve into the purpose of our research and the objectives for this project.

1.1 The Sport of Tennis

Tennis is a racket sport that can either be played between 2 players, known as singles, or 4 players, known as doubles. While both singles and doubles are similar, this explanation will focus purely on singles tennis matches.

Any given match has one player acting as the server and the other as the returner. A match is made up of sets, games and points. A normal match consists of a best-of-three sets format. Grand Slams are the only tournaments that have a best-of-five sets format, namely the Australian Open, Rolland Garros, Wimbledon and the US Open.

Each set comprises of games, with a player needing 6 games to win a set. Players must win with a margin of at least 2 games over their opponent to take a set. For example, scorelines of 6-2, 6-4 or 7-5 would indicate that a set has been won. If the set score reaches 6-6, the players play what is called a tiebreak, where they play first to 7 points and must again win by 2.

Tennis also has a unique scoring system within games. To win a game you need to win 4 points, with a winning margin of two more points than the opponent. The scoring system is 15, 30, 40 for the first, second and third points a player wins respectively. If a player is serving and it is 40-30 in their favour (the server's score is always given first) and they win the point, then they win the game. However, if a game reaches 40-40 it is called deuce, after which the game continues until one player wins 2 points in a row. Winning the first point after deuce is called advantage and winning the second point wins the game.

It should be noted that players alternate between being server and returner after every game. If a player is serving and wins the game, we say they held serve. Conversely, if they lost the game, we say their serve was broken. Therefore, if a player is serving and within one point of losing the game, they are facing what is called a break point. On the other hand, this is considered a break point opportunity for the receiver.

Finally, there is some terminology with respect to match play that must be clarified. If a player hits a serve that is unreturned and not touched, this is called an ace. If a player misses their first serve, called a fault, they are allowed to hit a second serve. Missing both serves constitutes what is known as a double fault. During a point, if a player hits a shot that is not touched, we refer to this as a winner; while missing a shot under no significant pressure is called an unforced error.

Now that the game of tennis and all the important terminology has been broadly defined, the next section will outline the research objectives of this project.

1.2 Research Objectives

This project sets out to investigate possible engineered features to accurately predict the outcome of male singles tennis matches on the ATP tour. Using various supervised learning techniques we aim to engineer different features that best capture the dynamics of a tennis match. The statistical learning methods used are logistic regression, classification trees, random forests and gradient boosting. These engineered features use past match statistics to predict future player performance.

This project begins by reviewing past literature on analytics in sports, tennis and the statistical theory that we will be implementing (Chapter 2). After this our data source and the Feature Engineering process will be fully explained (Chapter 3). This chapter looks at an historical averaging process, the difference between raw and engineered features as well as how these features were created. The models from above will then be reviewed in detail (Chapter 4). This is followed by a discussion of the application of the models and their results (Chapter 5). Finally, in the conclusion (Chapter 6), we will present key findings, look towards future work and give recommendations as to how our project and tennis match outcome prediction in general can be improved.

Chapter 2

Literature Review

Before we begin to predict tennis match outcomes, we must first review the relevant literature. For this project the literature review was broadly split into two categories, sports analytics and statistical theory. The sports analytics section will cover the application of statistical learning techniques in sport, while the statistical theory section will review literature on the relevant models that we are using in our analysis.

2.1 Sports Analytics

Analytics has a long history in sports and can be dated as far back as January 17, 1861. On this day the "Beadle's Dime Base Ball Player" compendium was published, and it states that one should look at both a player's batting and fielding statistics to determine their baseball skill level (The Video Analyst, 2020). However, the explosion of sports analytics has only come in recent times. There are two major categories of sports analytics that this project will discuss. These are analytics relating to player and game performance, as well as outcome prediction in sports.

2.1.1 Analytics in Player and Game Performance

Most fans associate sports analytics with the enhancement of team performance. Team performance is improved by being able to select the best possible team, choose optimal strategies and making good decisions on the field or court during a game (Davenport, 2014).

The most famous example of this is most likely Moneyball (Lewis, 2004), which outlined how the Oakland Athletics baseball team used an analytical approach to draft players to their Major League Baseball team. Players were selected based on proven performance measures, such as how often they get on base while batting (Davenport, 2014).

However, even years prior to Moneyball analytics was used by the Dallas Cowboys to win two Super Bowls in the 1960s, thanks to a computer programmer and statistician by the name of Salam Qureishi (The Video Analyst, 2020). Another recent example of the power of analytics in sports is the paper by Sarlis and Tjortjis (2020). This project outlines how we can evaluate the performance of basketball players and teams using machine learning and data mining techniques.

2.1.2 Predictive Analytics in Sports

While assessing player performance is useful, being able to accurately predict the outcome of sports matches is arguably more powerful. This can be done for any sport and there is a plethora of available literature. Here, we expand upon past research examples.

Leung and Joseph (2014) wrote a paper on predicting the outcome of American college football matches. They used an unconventional approach by predicting the winner based on historical results of games between similar teams, as opposed to using historical data of the two competing teams.

Eryarsoy and Delen (2019) used the Naive Bayes method, decisions trees and ensemble methods such as random forests and gradient boosting, to predict the results of football matches in the Turkish Super League. A similar approach has been taken in other sports. Valero (2016) used artificial neural networks, decision trees and support vector machines to predict win-loss outcomes in Major League Baseball matches, and also to assess what features can lead a baseball team to victory.

This was also done by Pathak and Wadhwa (2016) to predict One Day International (ODI) cricket matches. They applied the Naive Bayes method, support vector machines and random forest methods to predict match outcomes. Based on these models and their results, they created a cricket outcome predictor which outputs the win/loss probability for any ODI match.

Finally, analytical methods do not necessarily need to be used for match outcome prediction. A paper by Gruetzemacher, Gupta, and Wilkerson (2016) talks about sports injuries and their prevention. In the paper, a design of an application is outlined that can be used for sports injury prevention screening. This is accessible to those with limited funding. Through this research one can potentially prevent long-term injuries to sports athletes.

2.1.3 Racket Sports

One sport type that is yet to be discussed is racket sports. Racket sports usually involve 2 or 4 players. Since there are a small number of players involved, it is easier to analyse visual data of a match. Some examples of this are given here.

Ye et al. (2020) present Shuttle Space in their paper, which is an analytics system used to analyse trajectory data of players and shuttles in badminton. Similarly, Judd, Wu, and Taylor (2014) processed images of squash matches to perform analysis of player movement and create heatmaps. From these images they analysed average location, average distance traveled and average speed of a player on a squash court.

McGarry (1993) also did research in squash. By analysing data from the 1988 Canadian Men's Open, he tested the hypothesis that consistent patterns of play at the highest level of squash provide a prediction of future performance or play patterns. This was done using a Markov model, which found insufficient evidence to support this hypothesis.

Finally, Wang et al. (2019) used a hybrid second order Markov chain to model rallies within a table tennis match. Furthermore, they introduced a visual analytics system which can be used to look at different players and their tactics based on previous matches. This enabled better preparation for future matches against these players. Their methods can also be applied to other racket sports such as tennis, which will be our focus.

2.1.4 Tennis

Most past research in tennis focuses on match outcome prediction. For example, Barnett and Clarke (2005) combine player statistics to predict the outcome of a tennis match. Similarly, Katić et al. (2011) assess the impact of match statistics on outcome predictions for Wimbledon and Roland Garros in 2009.

On the other hand, one could also take a slightly different approach. Usually these papers fit a small subset of supervised learning algorithms to various feature sets. However, in her paper titled "Searching for the GOAT of tennis win prediction", Kovalchik (2016) is attempting to find the best model amongst 11 chosen models. Her paper is less concerned with finding the best feature set, rather selecting the model with the most accurate match outcome predictions.

Feature Engineering is also key to our project. One way to engineer features is to difference statistics between players (Cornman, Spellman, and Wright, 2017). This aids in comparing between players and as Cornman, Spellman, and Wright (2017) mention, it achieves symmetry within the data. Symmetry means it does not matter which player we label as Player A or Player B. One could also difference complementary statistics between players, such as comparing Player A's serve to Player B's return, as was done by Sipko and Knottenbelt (2015).

This is similar to the approach of Barnett and Clarke (2005), who compare serve and return statistics but also incorporate the tournament average in their analysis. Sipko and Knottenbelt (2015) also mention historical averaging and time discounting, due to the fact that recent matches hold more value. This is again common practice, as it is illogical to predict the outcome of a match using the statistics of that match. It is possible that Ma et al. (2013) made this logic error. They achieved a prediction accuracy of 90.6% when using a logistic regression model, which is much higher than any other research paper we found. Cornman, Spellman, and Wright (2017) only achieved a 69.9% 5-fold cross-validation accuracy with logistic regression. Similarly, Wilkens (2020) only achieved a prediction accuracy of 68.9% when using a logistic regression model, a much poorer result.

A number of different supervised learning algorithms are used in tennis match outcome prediction, or any other research relating to tennis for that matter. The most common algorithms are logistic regression, support vector machines, random forest and neural network models, as applied by Cornman, Spellman, and Wright (2017). Wilkens (2020) used these four algorithms as well as a gradient boosting approach. Sipko and Knottenbelt (2015) also used a logistic regression and support vector machines, along with an artificial neural network and a Markov chain model.

In this project we will use logistic regression, classification tree, random forest and gradient boosting models. These will be discussed in the next section of this literature review.

2.2 Statistical Theory

In this section we will review literature on the models used in this project. For detailed discussions on the methodology of the models, see section 4.

2.2.1 Logistic Regression

Logistic regression is used to model a binary outcome. Cramer (2002) performs an excellent review on the origins of logistic regression. Logistic regression is based on the logit function, which evolved from the logistic function. The logistic function first originated in the nineteenth century, through the works of Pierre-François Verhulst. It was much later in 1922 when Pearl and Reed rediscovered the logistic function and predominantly used it to model population growth (Cramer, 2002). The term logit was first coined by Berkson in 1944 and describes what we now commonly refer to as the log-odds. This is just the inverse of the logistic function.

Most research papers on tennis use a logistic regression, as this is the most common and

well-known method to model a binary outcome. Examples of papers that use a logistic regression in tennis match outcome prediction include the works of Cornman, Spellman, and Wright (2017), Sipko and Knottenbelt (2015) and Ma et al. (2013).

2.2.2 Classification Trees

A 50 year review of decision trees performed Loh (2014) states that Morgan & Sonquist published the first paper on regression trees in 1963. In 1972 Messenger & Mandell extended the idea of regression trees to categorical variables, to formulate classification trees. Classification trees are important, due to the fact that they are the building blocks for random forest and gradient boosting models. For this reason none of the literature we reviewed used decision trees to predict match outcomes, since random forest and gradient boosting methods employ similar ideas.

2.2.3 Random Forest

A random forest is an ensemble method that can be used for regression or classification and it was developed by Breiman in 2001 (Fawagreh, Gaber, and Elyan, 2014). Extensions of his work have been done. For example, Tsymbal, Pechenizkiy, and Cunningham (2006) found a way to improve the accuracy of random forests on certain datasets, by moving away from the majority voting technique. Amaratunga, Cabrera, and Lee (2008) investigated cases where random forests perform worse, as these cases contained too many features with very few informative ones.

A random forest is quite a common algorithm to use in tennis match outcome prediction and was applied by papers such as Cornman, Spellman, and Wright (2017) and Wilkens (2020).

2.2.4 Gradient Boosting

The two most popular variations of boosting are AdaBoost and gradient boosting. According to a lecture given by Hastie (2003), the idea of boosting first appeared in papers written by Schapire (1990) and Freund (1995), before the two authors collaborated in 1995 to formalise what is known as AdaBoost. Gradient boosting is a more recent algorithm, with the first implementation performed by J. H. Friedman in 2001. Friedman's ideas are further explained in "The Elements of Statistical Learning" (J. Friedman, Hastie, and Tibshirani, 2001). Wilkens (2020) used gradient boosting for prediction of tennis match outcomes; however, boosting is scarcely used in comparison to other algorithms.

2.3 Chapter Summary

Analytics has a broad range of application in sports. Predictive analytics can be applied to many different sports, for example baseball, basketball, racket sports and especially tennis. Most past literature focuses on the prediction of match outcomes using historical data. While feature engineering is mentioned in many papers, more emphasise is usually placed on the specific models used. Features are often averaged over past matches and differenced between players to create meaning, rather than specifically modelling the dynamics of a tennis match.

We will be using the following supervised learning algorithms: logistic regression, classification trees, random forests and gradient boosting models. These supervised learning techniques have origins spanning the 20^{th} century. Appearing often in literature, these models are applied to a wide array of classification problems and most are commonly used when predicting the outcome of tennis matches.

Chapter 3

Feature Engineering

While the accurate prediction of tennis match outcomes is a key research objective of this project, equally as important is the engineering of predictive features. These features are used by the supervised learning algorithms to predict the winner of a tennis match. The features are created using data that includes various statistics surrounding tennis, and this chapter will discuss the data source, historical averaging as well as raw and engineered features.

3.1 Data

3.1.1 Data Source

Before we can begin engineering features, we need to retrieve the data that will be manipulated to create them. Tennis is a sport that is well suited to analytical techniques, as many types of data are widely available. All information regarding past games, tournaments and players can be found on the ATP website. However, many data sources compile the data listed on this site into easily manipulated CSV files. This format is well suited to the needs of this project, and we have decided to make use of one of these sources. The data provided by Lin and Juko (2020) gives detailed data web-scraped from the ATP website. It is this source that will be used to make tennis match outcome predictions. The data is split into 3 categories:

- Player statistics - details about each player such as their age and height.
- Tournament statistics - information such as when the tournament was played and on which surface.
- Match Statistics - data regarding player performance in each match.

While all types of data are important when predicting match outcomes, match statistics contain the most useful information. They provide an historical record of how a player performs in each match that they play. This includes many statistics, as can be seen in table 3.1 below.

The tournament statistics are not explicitly used to predict match outcomes. However, the start date of each tournament, as well as the court surface of the tournament, do play a vital role in this analysis. The dates allow for historical averaging, as discussed in section 3.1.3 below, which is critical for predicting a match before it is played. Court surface is also extremely important, as this can impact the match. It can be said that some players perform better on certain court surfaces, and taking this into account could strengthen the analysis.

Unfortunately, Lin and Juko (2020) did not provide sufficient player statistics. Instead, we web scraped these ourselves from the ATP site using R. The datasets were then merged, resulting in a large dataset that we could begin to manipulate.

Table 3.1: List of Match, Player & Tournament Statistics

Match Statistics	Player Statistics	Tournament Statistics
Duration	Age	Tournament Id
Match Order	Weight	Tournament Date
Winner	Height	Tournament Conditions
Player A & B Names	Date of Birth	Tournament Surface
Player A & B Aces	Dominant Hand	Prize Currency
Player A & B Double Faults	Back Hand Type	Tournament Location
Player A & B First Serves In	Year Turned Pro	
Player A & B First Serves Total		
Player A & B First Serve Points Won		
Player A & B First Serve Points Total		
Player A & B Second Serve Points Won		
Player A & B Second Serve Points Total		
Player A & B Break Points Saved		
Player A & B Break Points Serve Total		
Player A & B Service Games Played		
Player A & B First Serve Return Won		
Player A & B First Serve Return Total		
Player A & B Second Serve Return Won		
Player A & B Second Serve Return Total		
Player A & B Break Points Converted		
Player A & B Break Points Return Total		
Player A & B Return Games Played		
Player A & B Service Points Won		
Player A & B Service Points Total		
Player A & B Return Points Won		
Player A & B Return Points Total		
Player A & B Total Points Won		
Player A & B Total Points Total		
Player A & B Sets Won		
Player A & B Games Won		
Player A & B Tiebreaks Won		
Player A & B Seed		

3.1.2 Data Cleaning

After retrieving the data to be used in this analysis, it is vital that the data is formatted and free of errors. Data cleaning is the act of "preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated or improperly formatted" (Sisense, 2020).

While the data provided is mostly well formatted, there are a few errors and some irrelevant information. During this process, all qualifying matches (matches played before a tournament to allow players to earn a place in the event) were removed. This project will only focus on tournament play. Matches that did not contain all information were also removed, as well as those with errors. Data that is irrelevant to the prediction of tennis match outcomes, such as the currency of prize money, was also removed.

After completing this process we were left with 26 259 matches between 773 unique players. The data consists of 3 court surfaces, namely hard, grass and clay. As can be seen in figure 3.1, there are far more matches played on a hard court than on any other surface. Very few

matches are played on grass, which may affect the analysis. Matches took place between the years of 2010 and 2019. There were an approximately equal number of matches per year, with some slight variation. We split this into a training and a test set. Matches played between 2010 and 2018 were used to train the models, while matches in 2019 were used as the test set.

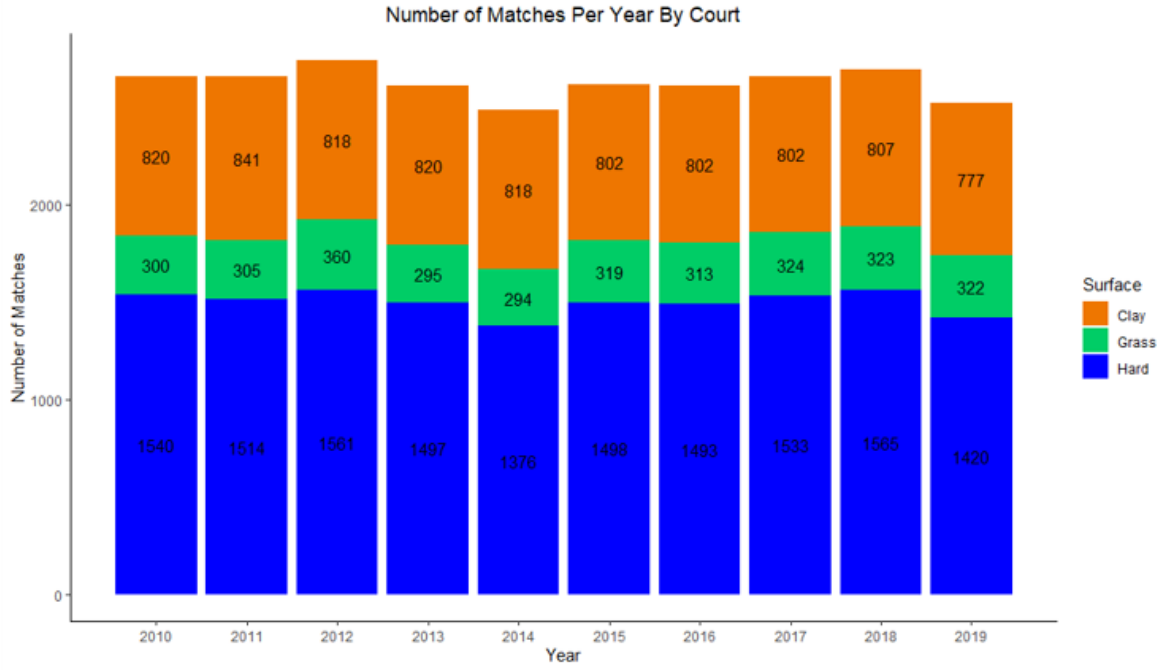


Figure 3.1: Number of Matches By Court per Year

3.1.3 Historical Averaging

The structure of the match statistics is such that they are given for each match after it has taken place. When predicting the outcome of a tennis match, it is rare that we will have the statistics of the match that we are trying to predict. In the real world, it is also of little use to predict the winner of a match after it has taken place. What is more logical, and more interesting, is that we wish to predict the outcome of a tennis match knowing only the players and their seed. Seed is determined before tournament play, thus the seeds are known prior to the match we are predicting. Due to the fact that we are predicting a tennis match before it is played, a measure of past performance needs to be created. We need to know the historical performance of each player in order to predict the winner of a specific match.

The historical performance measure is created through historical averaging. For each individual player, the average of every match statistic in the past 24 months is calculated. This is done in R, which iteratively takes each player in a match, finds all of their past matches in the specified time period, and calculates the averages leading up to the match. The historical average is then found for every match in the dataset. This ensures each match is associated with a recent average of both players' performance, to enable the supervised learning models to create match outcome predictions.

This process is repeated but stratified by the court surface each match is played on. Thus, if the match we are attempting to predict is played on grass, we only look at matches in the past 24 months that were also played on a grass surface. This was repeated for each match, leaving us with both an historical average and an historical average by court. These are also referred to as the rolling average and the rolling average by court in this project.

A problem this historical averaging creates is that when a player first appears in the dataset,

they will be assigned average values of 0. This is due to the fact that they have never played a match before, thus there is no historical record of their performance. While some models are robust enough to handle this, it must be considered during the prediction process.

3.1.4 Time Discounted Historical Averaging

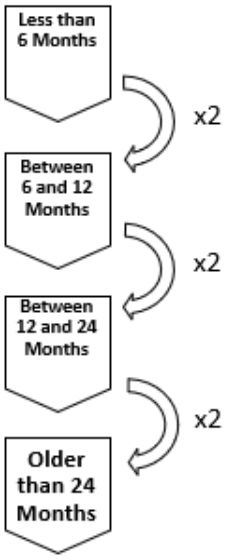


Figure 3.2: Time Discounting Illustration

In order to make accurate predictions, it may also be useful to introduce a recency bias into our historical average. By giving recent matches a higher weighting when calculating the historical average, we are able to capture a player's recent performance. If a player has been playing particularly well of late, this ensures that their improvement is reflected.

The time discounted historical average is created using a weighting technique. This is similar to what was done by Devcic (2020), where data points from certain time periods are weighted more than data from older periods. In this application, data is broken up into matches in the last 6 months, matches between 6 and 12 months ago, matches from 12 to 24 months ago and matches older than 24 months.

Data in the most recent time period is weighted double that of data from the next most recent. This is then in turn weighted double the data from the period after, as is illustrated in figure 3.2. It creates a feature set that will hopefully accurately represent how a player is likely to perform in a given match. A possible exception to this would be if there is no data in a time period. If this is the case, the time period is ignored and data is worth double the time period after the one that is left out.

Once again, we repeat this process stratifying by court surface. We only consider matches played on the same court surface as the match we are attempting to predict, and repeat the same weighting process. These are also referred to as the weighted rolling average and the weighted rolling average by court in this project.

3.2 Raw Features

We define raw features simply as the historically averaged match statistics given to us in the data source, as outlined in table 3.1. They are simply an historical record of a player's past performance, and in no way attempt to model the dynamics of a tennis match. No data is combined or manipulated, and we predict tennis match outcomes using the simplest data possible.

These performance statistics are then split in to feature sets according to the type of historical averaging used. Thus, we are left with four sets of raw features. The first and second of these feature sets are calculated using historical averaging and historical averaging by court respectively. The third and fourth are calculated using time discounted historical averaging and time discounted historical averaging by court. We will fit each model, as outlined in section 4 below, to each of these feature sets.

3.3 Engineered Features

Engineering features is a major part of this analysis. We hope to take the raw features outlined above, and manipulate them in order to create better match outcome predictions. These features will hopefully capture how a tennis match occurs in reality. There are a few vital components to a tennis match that we will try to replicate by engineering these features. These involve modelling key moments in tennis matches, comparing the players involved in the match as well as adjusting certain raw features that are dependent on match length.

3.3.1 Serve and Return Features

At its simplest, the two key facets of a tennis match are serving and returning. In any given tennis match, one player acts as the server and the other as the returner, alternating after every game. For this reason, we would like to engineer a feature that takes serving and returning into account when predicting tennis match outcomes.

Cornman, Spellman, and Wright (2017) directly compared the serve of one player to their opponent, repeating this for the return. We believe this does not capture the fundamental serve-return dynamic in tennis. If a player has a strong serve, they are not necessarily going to hold serve easily. This would only be the case if their opponent has a weak return.

Instead of a direct serve-to-serve comparison, we believe a better feature would compare a player's serve to their opponent's return. This was first pioneered by Barnett and Clarke (2005) as a way to model the dynamics of a tennis match. We aim to recreate this feature in a similar manner, by comparing the serve of Player A to the return of Player B and vice versa.

This comparison is made using two metrics, the winning percentage on serve (WPS) and winning percentage on return (WPR) for both players. The formulation of the WPS and WPR are shown below:

$$\text{WPS} = \frac{\text{Service Points Won}}{\text{Service Points Total}} \times 100$$

$$\text{WPR} = \frac{\text{Return Points Won}}{\text{Return Points Total}} \times 100$$

This was calculated for every match a player participates in and is historically averaged using the different measures outlined in 3.1.3 above. From these, we create two "Serve Advantage" (SA) features. This is a simple subtraction of Player B's historical WPR from Player A's historical WPS, and vice versa. Differencing these percentages indicates how a player's serve compares to their opponent's return. This tries to model the serve-return dynamic to show if a player has an advantage on their serve.

$$\begin{aligned} \text{SA}_A &= \text{WPS}_A - \text{WPR}_B \\ \text{SA}_B &= \text{WPS}_B - \text{WPR}_A \end{aligned}$$

We can then combine these two features by subtracting them, to create an overall serve advantage feature. This attempts to model any advantage a player may have due to the strength of their serve or return. We hope that the overall serve advantage will be able to predict the outcome of a tennis match with precision.

$$\text{Overall serve advantage} = SA_A - SA_B$$

The final serve and return feature that we created was 'completeness', similar to what was done by Sipko and Knottenbelt (2015). Completeness is a measure of the overall strength of a player and is found by multiplying a player's WPS by their WPR. This is repeated for

both players in a tennis match. A high value for this figure will indicate that a player has both a strong serve and a strong return, showing the overall strength of a player.

$$\text{Completeness}_A = \text{WPS}_A \times \text{WPR}_A$$
$$\text{Completeness}_B = \text{WPS}_B \times \text{WPR}_B$$

3.3.2 Break Point Features

Another vital component of tennis matches that we would like to model is break points. Break points are key moments in any tennis match. As mentioned in section 1.1, in order to win a set a player must have a margin of at least 2 games. For this to occur, a player must 'break' their opponents serve, while not letting their serve be broken. This is not as easy as it appears, due to the fact that the serving player has a major advantage. Some past literature include break points in their analysis; however, simply as a raw feature such as was done by Sipko and Knottenbelt (2015). We would like to expand on this, and try to capture the dynamics of a tennis match using break points. Thus, the engineered features in this section attempt to capture the break point component of a tennis match by looking at the frequency of break points, as well as the conversion rate.

Frequency Features

First, we look at the frequency of break points. A player that is creating many break points may be a strong returning player. If many break points are created by a player in a particular match, it indicates a strong advantage to that player. However, we cannot simply look at the number of break points created. Every match is unique, and the number of games played within them will be different. Thus, we define the frequency that break points are created as:

$$\text{Frequency Created} = \frac{\text{Break Points Created}}{\text{Return Games Played}}$$

On the other hand, a player can also face break points on their serve. A player that is facing many break points may be a weak serving player. This would indicate an advantage for their opponent. Once again, this cannot be a simple figure and needs to be adjusted for match length. Thus, the frequency of break points faced is given by:

$$\text{Frequency Faced} = \frac{\text{Break Points Faced}}{\text{Service Games Played}}$$

These break point frequency features are created for every match. We then use both the historical averaging and the time discounted historical averaging techniques from above. This ensures we have a record of how often a player creates and faces break points, to hopefully aid in the prediction of tennis match outcomes.

Conversion Features

Equally as important as the frequency of break points, is the ability of the player to convert the break points that they create or save those that they face. Being able to perform when under pressure is vital for a tennis player. This is often referred to as 'big match temperament', the ability of a player to perform well in critical moments. It is this temperament that these conversion features aim to capture.

The first measure we define is the conversion rate of break points created. This measures how well a player is able to take advantage of the break points they create. Creating a break

point is not how a tennis match is won, only the break points that are converted give a player an advantage. Thus we define the conversion rate as:

$$\text{Conversion Rate} = \frac{\text{Break Points Converted}}{\text{Break Points Created}}$$

Furthermore, it is equally as important for a player to remain calm under pressure and save the break points that they face. While a player may face many break points, they can still save these and not have their serve be broken. This is defined below and attempts to give an indication of how well a player is performing under pressure on their own serve.

$$\text{Saving Rate} = \frac{\text{Break Points Saved}}{\text{Break Points Faced}}$$

We again use historical averaging and time discounted historical averaging for the conversion features. This ensures we have a past record for each player in the match we are trying to predict.

Break Point Comparison

Now that we have these features, we still need to compare the two players in the match we are trying to predict. This is done by taking the difference of each feature. If there is a large difference between the values for each player, it would indicate that one player in the match has an advantage over the other. This leaves us with 4 final break point features to try and determine the winner of a tennis match:

1. Player A Frequency Created - Player B Frequency Created
2. Player A Frequency Faced - Player B Frequency Faced
3. Player A Saving Rate - Player B Saving Rate
4. Player A Conversion Rate - Player B Conversion Rate

3.3.3 Comparison Features

When predicting tennis match outcomes, we are making a binary classification choice. Thus, it can be useful to directly compare the players involved. While this concept has already been used in sections 3.3.1 and 3.3.2, this is carried over to other raw features. This idea is similar to what was done by Cornman, Spellman, and Wright (2017), as explained in section 2.1.4. In addition to differencing the above engineered features, we also wish to directly compare the seeding and the average match duration.

Seeding is used to separate the top players in a tournament. A low seeding indicates that a player is highly ranked. The maximum seeding in a tennis tournament is 33, however, this is a special case caused by player withdrawals. Seeds only capture the top 32 ranked players participating in a tournament. For this reason we assigned players that were unseeded as seed 34, a number higher than the maximum. When comparing players, this ensures that the feature is a large number when a player with a low seed faces an unseeded player. We hope that this will capture the general skill level between the players in the match.

$$\text{Seeding Difference} = \text{Seed}_A \text{ - } \text{Seed}_B$$

We also introduce a comparison of the average match duration for each player. Tennis matches are intense, thus players with a shorter average duration may be less fatigued during the match. Players that play shorter matches may also be more clinical since they spend less time on the court. This comparison is also done by differencing the average match duration for both players.

$$\text{AMD} = \text{Average Match Duration}$$
$$\text{Duration Difference} = \text{AMD}_A \text{ - } \text{AMD}_B$$

3.3.4 Adjusting Features

Many of our raw features do not account for factors such as match length or number of games played. For this reason we need to manipulate these features to be consistent. For example, it is incorrect to only use the raw number of aces, as this is dependent on the total amount of serves. This is also true for double faults. A higher number of second serves gives more opportunities for a double fault to occur, thus these features must be adjusted.

$$\text{Aces Percentage} = 100 \times \left(\frac{\text{Total Aces}}{\text{Total Serves}}\right)$$

$$\text{Double Fault Percentage} = 100 \times \left(\frac{\text{Total Double Faults}}{\text{Total Second Serves}}\right)$$

3.3.5 Head-to-head Feature

Something else that is important when predicting the outcome of tennis matches is the past results between the players of interest. It is possible to isolate all the matches between the two players in the match we are predicting, and this could give a good indication of how the match will play out. For example, if two players have met three times and Player A has won all three, it indicates that Player A likely has some advantage over Player B. This takes the form of a percentage and is given in terms of Player A. Thus, if Player A wins 1 game out of three, the record will show 33.33%.

We can extend this concept further, and calculate a head-to-head record on each court surface. Some players may have an advantage on certain courts, and stratifying the head-to-head record by court accounts for this.

However, often players are meeting for the first time. In these cases, we assign the the head-to-head record as 50%. This is implies that either player has equal chance of winning the match.

This is not without flaws. One major issue is that the number of matches is not reflected. If two players meet often, and a player has won all of these games, this implies a greater advantage than if the players have only met once. However, in both cases the record will be either 100% if Player A always wins, or 0% if Player B always wins. This could be adjusted in future work; however, we believe the impact will be minimal when predicting tennis match outcomes.

3.4 Final Predictive Datasets

Now that we have created the historical averages, time discounted averages as well as raw and engineered features, it is useful to understand what the final predictive datasets entail. In the models that are explained in section 4, we will fit each model to 8 different datasets. Four of these datasets are raw features, namely raw features using an historical average, historical average by court, time discounted historical average and time discounted historical average by court.

In the datasets, each row is a match and every column is a variable. The first 11 columns contain information pertaining to the current match. These are the match ID, the players in the match and the winner. Also included is tournament information. This is tournament ID, name, type, date, conditions, surface and the order of the match within the tournament. Each row then contains the match statistics found using the historical averaging method of

that dataset. For example, a column contains the historical average of the first serve percentage for Player A. Each row in this column contains the average first serve percentage value up until the particular match was played. This enables us to predict the match using past statistics, as they are stored in the relevant row.

There are also four engineered feature datasets. These take a similar form, with each dataset using a different historical averaging technique. Once again, each row contains a match, where the first eleven columns remain the same. The other columns contain the historical record for the features that we have created, as outlined above.

Using these eight datasets, we are able to predict match outcomes using various supervised learning techniques. We are then able to compare not only the model performance, but also how the engineered features perform against the raw features. We can also compare how the different historical averaging methods affect the predictive power. This ensures we can perform the best possible analysis and determine the optimal method of predicting tennis match outcomes.

3.5 Chapter Summary

The data for this project was sourced from a GitHub repository by Lin and Juko (2020). All the match statistics provided by Lin and Juko (2020) were historically averaged using various techniques. These included simply averaging the matches in the past 24 months, as well as introducing a recency bias. These historically averaged and time discounted historically averaged feature sets were also stratified by court, to give us four raw feature sets.

An important part of this project is feature engineering. We engineered a serve and return feature, comparing the serve of one player to the return of their opponent, to capture the serve-return dynamic within tennis matches. We also considered the frequency of creation and conversion rate of break points, to see how much pressure a player is put under and how they handle this pressure. Some features, such as aces and double faults, were adjusted to make them more meaningful. A head-to-head and a head-to-head by court feature were also created to account for past records between two players. These engineered features were again historically averaged, time discounted and stratified by court, to give us four engineered feature sets.

Chapter 4

Methodology

When predicting the outcome of tennis matches, this project utilises various supervised learning techniques. These techniques take the data and predictive features outlined in section 3 above, to create tennis match outcome predictions. This chapter details the methodology behind these techniques and how they are used for this application.

4.1 Logistic Regression

Logistic regression is used to model a binary outcome variable Y, which in this case is the winner of a tennis match. As opposed to modelling the response variable directly, the logistic regression models the probability that Player A wins a particular match. If Player A wins the match, $Y = A$. If Player B wins the match, $Y = B$. We are interested in determining the probability of $Y = A$ being the outcome given a set of linear predictors, such as serve advantage or average first serve percentage in past matches. This could be specified as the conditional probability below, where the matrix $\mathbf{X}$ represents our features:

$$p_A = P(Y = A|\boldsymbol{X}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

The issue is that this linear set of predictors can predict values of $Y < 0$ or $Y > 0$. This does not make sense in the context of probabilities (James et al., 2013). We therefore need a function which guarantees values between 0 and 1. In this case we use the logistic function:

$$p_A = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_n x_n}}{1 + e^{\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_n x_n}}$$

After some manipulation we end up with:

$$\frac{p_A}{1 - p_A} = e^{\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_n x_n}$$

The quantity $\frac{p_A}{1-p_A}$ is referred to as the odds, or odds ratio, of Player A winning and can take on any value between 0 and ∞. We can therefore take the natural logarithm of both sides of the above equation. This gives us what is called the logit function:

$$log\left(\frac{p_A}{1 - p_A}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

The logit function is a logarithm of the odds ratio, called the log-odds. This is a sum of our features weighted by their respective coefficients, which can easily be back transformed to get a predicted probability as shown above. We are now able to get sensible predicted probabilities. Ultimately, we want to be able to classify observations into the categories of Player A or Player B based on these probabilities.

In order to do this we need to find some threshold. If the probability is above this threshold, we would predict Player A as the winner. If the probability is blow the threshold, we would predict Player B as the winner. The most commonly used threshold is 0.5 but this is not always the optimal threshold.

There is a robust method of choosing this threshold, by making use of a confusion matrix. To explain this further, an example of a confusion matrix for the tennis context is given below:

Table 4.1: Confusion matrix

	Observed	
Predicted	Player A	Player B
Player A	a	b
Player B	c	d

We would like to accurately predict the cases where Player A wins the match (true positives) and those where Player B wins the match (true negatives). In doing so, this leads to minimising incorrect predictions (false positives and false negatives). The proportion of those matches where Player A wins and we make a correct prediction is called sensitivity, defined as $\frac{a}{a+c}$. Conversely, the proportion of those matches where Player B wins the match and we correctly predict as such is called specificity, defined as $\frac{d}{b+d}$. The optimal threshold maximises both sensitivity and specificity. This optimal threshold is determined using the receiver operating characteristics (ROC) curve (James et al., 2013), which will be discussed in section 5.1.1.

4.2 Classification Trees

The aim of classification trees, a branch of decision trees, is to determine a set of rules to make predictions on a categorical dependent variable. Classification trees use a set of predictive variables, the features from section 3, to predict whether Player A or Player B wins a match. Collectively, these are called the feature space. Mathematically, the set of predictive variables is represented by $X_1, ..., X_p$ and the categorical variable by Y_i.

In order to create the set of rules, we begin by partitioning the feature space (James et al., 2013). Partitioning the feature space means to group observations based on the value for a single predictive variable. These partitions create regions in the feature space, $R_1, ..., R_j$, for every split on an X variable. For example, if we create the rule of $X_1 < k$ we will create 2 regions. R_1 will contain all the observations whose value of X_1 is less than k, while R_2 will contain observations that have a value of X_1 larger than or equal to k. This can be seen in figure 4.1 below.

Thus, our rule has partitioned the observations depending on the value of X_1. We can then predict the value Y for each region. The predicted value of an observation with $X \in R_j$ is given as the most commonly occurring value of Y_i in the region R_j.

After partitioning the feature space, we can define our actual classification tree as seen in figure 4.2. We again split at $X_1 < k$, and can now understand how these regions are formed. The regions are now called nodes, of which there can be a few types. First is the root node, which is the node at the top of the tree. This contains all the observations. In the figure 4.2 below, R_1 and R_2 are defined as the terminal nodes, which have no other nodes below them. In this case R_1 and R_2 are 'child' nodes of the root node. Furthermore, the root node is the

'parent' node of R_1 and R_2.

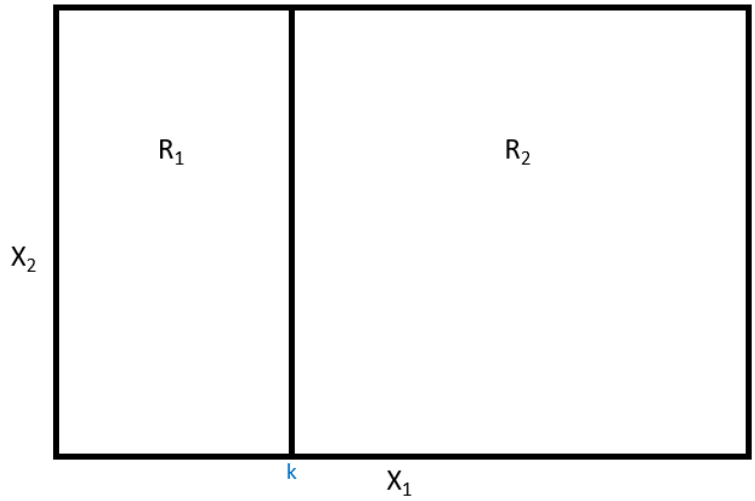


Figure 4.1: Feature Space Partition on X_1 at k

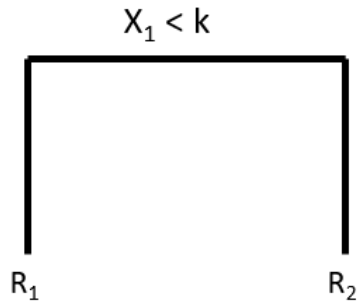


Figure 4.2: Tree Representation of Feature Space Partition on X_1 at k

However, how do we know where to partition the feature space to create a set of rules? In order to do this, we need to define a splitting criteria. Classification trees make use of 3 different splitting criteria, namely the Gini Index, Entropy and Deviance measures as outlined below. At each step we attempt to find a split point s on the predictive variable X_k, such that $R_1 = \{X|X_k < s\}$ and $R_2 = \{X|X_k \geq s\}$. We choose the split point s that leads to the greatest reduction in the chosen splitting criteria. We continue this until we reach a stopping condition. This condition is defined by the researcher and may be a minimum number of observations in a region or a minimum reduction in the value of the splitting criteria. This process is called recursive binary splitting, and it provides a much less computationally heavy method for creating classification trees.

4.2.1 Gini Index

Ideally, each terminal node would include observations of only one response category (Britz and Lacerda, 2020c). This means that if the region R_j is created when $X_k < s$, all of the observations where $X_k < s$ have the same value for Y_i. This is the ideal scenario, where the terminal node is 'homogeneous'. The Gini index is a measure of how close to a homogeneous state a terminal node is. The Gini index takes the form:

$$G = \sum_{j=1}^{J} G_j \qquad G_j = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk})$$

Where $\hat{p}_{jk}$ is the proportion of observations in the response category k within leaf node j. G_j is the Gini index for node j, while G is the overall Gini index. A Gini index of 0 represents a homogeneous node, thus, the smaller the Gini index value the better.

4.2.2 Entropy

Entropy is another method that makes use of the same homogeneity concept. While similar in concept to the Gini index, it has a slightly different formulation. As opposed to multiplying $\hat{p}_{jk}$ by $1 - \hat{p}_{jk}$, we instead use the natural logarithm to calculate the entropy of each terminal node. We then sum each entropy to calculate an overall value for the tree. Thus, the entropy is given by:

$$E = -\sum_{j=1}^{J}\sum_{k=1}^{K}\hat{p}_{jk}log(\hat{p}_{jk})$$

Where we are again summing the proportion of each response category in every terminal node. Reducing this to 0 is again the ideal scenario, thus the split that produces the greatest reduction in entropy will be chosen.

4.2.3 Deviance

The last type of splitting criteria is deviance. The deviance is not a measure of homogeneity, but rather views the tree as a probability model and chooses the model that has highest likelihood (Britz and Lacerda, 2020c). This is equivalent to minimising the deviance, which takes the form:

$$D = -2\sum_{j=1}^{J}\sum_{k=1}^{k} n_{jk}\log(p_{jk})$$

As we can see, this takes into account the number of observations. It is the most similar measure to the residual sum of squares, and chooses the model that makes the data the most probable.

4.2.4 Classification Tree Pruning

The final step in our procedure is to prune the classification tree. In the process above, the tree was grown to a depth that was arbitrarily specified using a stopping condition. Good practise is to grow large trees, as larger trees are more likely to produce a model that classifies known observations well. However, these large trees are often too complex to make accurate predictions on unseen data, fitting a model too closely to the training observations. For this reason we must 'prune' the tree by reducing the number of terminal nodes, to make the tree less complex.

In order to perform this pruning, we can either use any of the above splitting criteria or the classification error of the tree to create a new measure called the cost complexity criteria. The cost complexity criteria is defined as:

$$C_\alpha(T) = C(T) + \alpha|T|$$

Where $|T|$ is the number of terminal nodes for a subtree T, α is a tuning parameter and $C(T)$ is either the chosen splitting criteria or the classification error. It is important to note that the pruning process is performed via cross-validation and not using the test dataset.

In cross-validation the data is randomly divided into K groups of equal size. A model is then fit to all groups except the k^{th} group. Predictions are then made using the k^{th} group,

and the cost complexity criteria is calculated. This is repeated for all the groups. Each cost complexity criteria is then combined from across all of these iterations.

For each value of α we try to find a subtree T that minimises the cost complexity criteria. This has the effect of comparing a tree with a different number of terminal nodes, by finding the cross validated value for $C_\alpha(T)$. As α increases from 0, this prunes the tree in a nested manner (Britz and Lacerda, 2020c). We can then decide only how much complexity is worth being introduced for a certain reduction in the cross-validation error.

4.3 Random Forest

A random forest is a supervised learning model that consists of a large number of individual classification trees (Yiu, 2019). This is extremely useful, as averaging a large number of predictions will lower the sampling variance compared to the predictions from a single training set.

In order to construct these ensembled forests, we use a technique called bootstrapping. Bootstrapping is the process of sampling observations at random with replacement (Britz and Lacerda, 2020a) from the training set. If we create B bootstrapped samples, we can then grow B unpruned trees using the process as in section 4.2. We then make B predictions and combine these to make an averaged and final prediction.

Unlike other tree ensemble methods, random forests attempt to minimise the correlation between the trees. This is done by fitting each tree using only a random sample of m predictors, where m is less than the total number of predictive variables, p. Thus, for each bootstrapped sample we select a random m predictive variables and build an unpruned tree. This has the effect of decorrelating the trees, as the trees will be different due to the subset of predictive variables.

Unlike singular classification trees, we do not need cross-validation to estimate the test error for our model. Due to the fact that we are sampling from our original observations, we can use observations outside of the sample to calculate an 'out of bag' error. The number of out-of-bag observations can be show to be approximately a third (Breiman, 2001). We are able to predict a response for these out-of-bag to observations for each B tree, to create the out-of-bag error rate. This emulates a test error. The out-of-bag error rate also enables us to ensure that the random forest uses enough decision trees. Once the rate is stable and with no drastic changes, the number of trees is large enough.

However, subsetting the predictive variables means it is difficult to interpret the random forest. It becomes unclear how the model is creating predictions with a large number of trees. For this reason we use a variable importance plot. This is created by recording the amount the splitting criterion is improved for each tree and taking the average over all B trees for each feature (Britz and Lacerda, 2020a). This lets us understand which features are most important when predicting the outcome of tennis matches and eases interpretation of the model.

It is clear that there are many hyperparameters that must be defined. Firstly, the value of m must be chosen. During classification applications, the default hyperparameter value of m is given by $\sqrt{p}$. We also need to define the number of trees in the random forest using the method above. Next, we must set the value for the minimum node size hyperparameter. The default for this is 1 in classification examples. Finally, we need to define the splitting criteria for each tree in the random forest. These are outlined in section 4.2. When using the ranger package in R, the splitting criteria must be the Gini index for classification. In order to find

optimal values for these hyperparameters, we use a grid search. Using the caret package, we are able to train many random forests with different hyperparameter values and determine the optimal model.

4.4 Gradient Boosting

Gradient boosting, like random forests, consists of a large number of individual classification trees. Despite also being an ensemble model, the overall approach is rather different. In gradient boosting we aggregate many weak learners into strong learners. Weak learners are small, uncomplicated trees that are aggregated by learning from mistakes of previous trees (Britz and Lacerda, 2020b). We fit new trees based on the residuals of the previous tree. This means the algorithm is sequential and successive trees are dependent.

To build the algorithm on the training dataset, we need to define an initial prediction. We are trying to predict the winner of a tennis match as either Player A or Player B, which means we are dealing with a classification problem. In this case we can predict in pretty much the same way as for logistic regression (section 4.1). We can use the logistic function where the initial odds will be the observed proportion of Player A divided by that of Player B. This is used to calculate an initial log of the odds and then back transformed to get a predicted probability p_A (section 4.1). The initial residuals are then $1 - p_A$ if the winner in the training dataset is Player A and $0 - p_A$ if the winner is Player B. With these initial residuals we are ready to fit trees.

We fit B trees, using d splits, that predict the new log-odds based on scaled residuals and a learning rate λ. The variable used to make each split is chosen using either the Gini index or the log loss function (Brownlee, 2016). We then update the predicted probability and can calculate new residuals. The steps are outlined below, based on the approach of StatQuest (2019):

1. Fit a tree based on the current residuals, r_i, of each observation to calculate new predicted log-odds for each terminal node, where n_l is the number of observations in a given terminal node and p_i is the previous predicted probability of that given observation.

$$\text{predicted log-odds} = \frac{\sum_{i=1}^{n_l} r_i}{\sum_{i=1}^{n_l} p_i(1-p_i)}$$

2. Add this predicted log-odds to the previous log-odds, weighted by lambda:

$$\text{new log-odds} = \text{previous log-odds} + \lambda(\text{predicted log-odds})$$

3. Calculate the new predicted probability:

$$\text{predicted probability} = \frac{e^{(\text{new log-odds})}}{1 + e^{(\text{new log-odds})}}$$

4. Calculate new residual:

$$\text{new residual} = 1 - \text{predicted probability, if the winner is Player A}$$
$$\text{new residual} = 0 - \text{predicted probability, if the winner is Player B}$$

5. Repeat steps 1-4 B number of times. We can then predict the log-odds of out of sample observations using the expression below (StatQuest, 2019) :

$$\text{final predicted log-odds} = \text{initial log-odds} + \lambda \sum_{b=1}^{B} \text{predicted log-odds}$$

We can convert the final predicted log-odds back to a probability (section 4.1) and then classify based on a threshold of 0.5. Here we do not aim to optimise the threshold, but instead tune the hyperparameters d and λ, as well as the number of trees B. This can be done through cross-validation, as we can trial different combinations of the hyperparameters and determine which combination gives the best cross-validated predictive accuracy.

As with random forests, the interpretation of the gradient boosting model is difficult. Therefore, we can again make use of variable importance plots to determine which variables are most important in predicting the outcome of a tennis match.

4.5 Chapter Summary

Four models will be used to predict the outcome of tennis matches. The logistic regression function is a binary classification algorithm that gives a probability of a player winning the match. Using a threshold, these probabilities are then converted in to a prediction, based on whether the probability is greater than the threshold or not. Classification trees create splits on individual variables, choosing the split that creates the largest reduction in a chosen splitting criteria. Classification trees can make use of 3 different splitting criteria, namely the Gini index, entropy or deviance. Random forests and gradient boosting are ensemble methods that use a large number of classification trees. Random forests use bootstrapping to create many trees, and aggregate the predictions to lower sampling variance. Random forests decorrelate the trees by only considering a subset of predictive variables for each bootstrapped sample. Gradient boosting sequentially aggregates many small, uncomplicated trees by modelling the residuals of the previous tree. All of these models may be useful when predicting the outcome of tennis matches, as they are proven to be applicable to classification problems such as ours.

Chapter 5

Application and Results

We would like to accurately predict the results of singles tennis matches on the ATP tour using an array of raw and engineered features. Having engineered the features, we can now make predictions for tournaments in the 2019 season, the test set. The analysis is first separated for the respective models before we compare the individual results. Models are compared using the classification rate on the test data.

5.1 Model Specific Results

5.1.1 Logistic Regression

To predict using logistic regression we need to determine the optimal threshold. As mentioned in section 4.1, we make use of a confusion matrix and the ROC curve to determine the optimal threshold value. Once we have this threshold we can make predictions on tennis data from 2019.

The ROC curve plots the false positive rate, $1 - \text{specificity}$, against the true positive rate, sensitivity. We would like to be as close to the top left corner as possible, as here we maximise sensitivity and specificity. The ROC curves can be plotted for all eight feature sets. Firstly, figure 5.1 shows the ROC curves for the raw feature sets. We can see that three of the four have similar predictive performance, but the historically averaged by court feature set performs a lot worse than the others.

The ROC curves for the engineered feature sets are shown in figure 5.2. Here the predictive performance is quite similar across all four feature sets.

However, the ROC curves will not show us which of the eight feature sets performs best at predicting the results of 2019 tennis matches. For this we use the classification rate. Table 5.1 shows the classification rate for all 8 feature sets on tournaments in 2019, and also gives the optimal threshold as determined from the ROC curve.

For the raw features, the time discounted historical averaging performs best with a classification rate of 62.58%. Whereas for the engineered feature sets, time discounted historical averaging by court performs best with a classification rate of 61.93%. The worst performing feature set is the raw historical average by court. Except for the historical average by court, the engineering of predictive features has not improved results significantly. In fact, the best performing feature set overall is the raw time discounted historical average feature set.

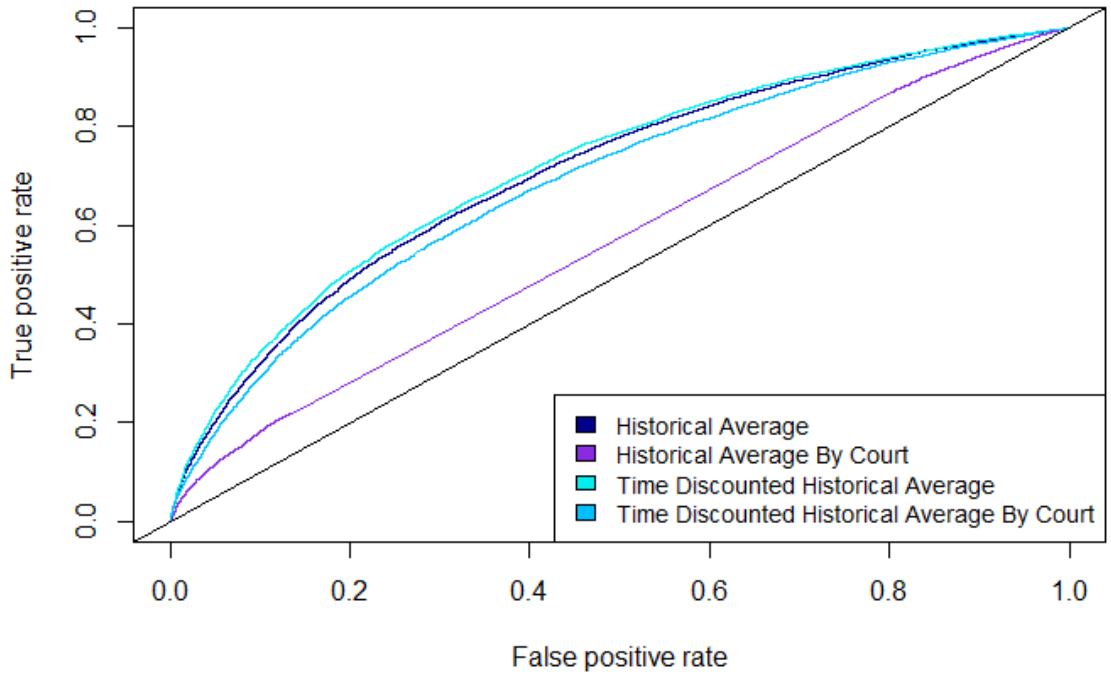


Figure 5.1: ROC curves for Raw Features

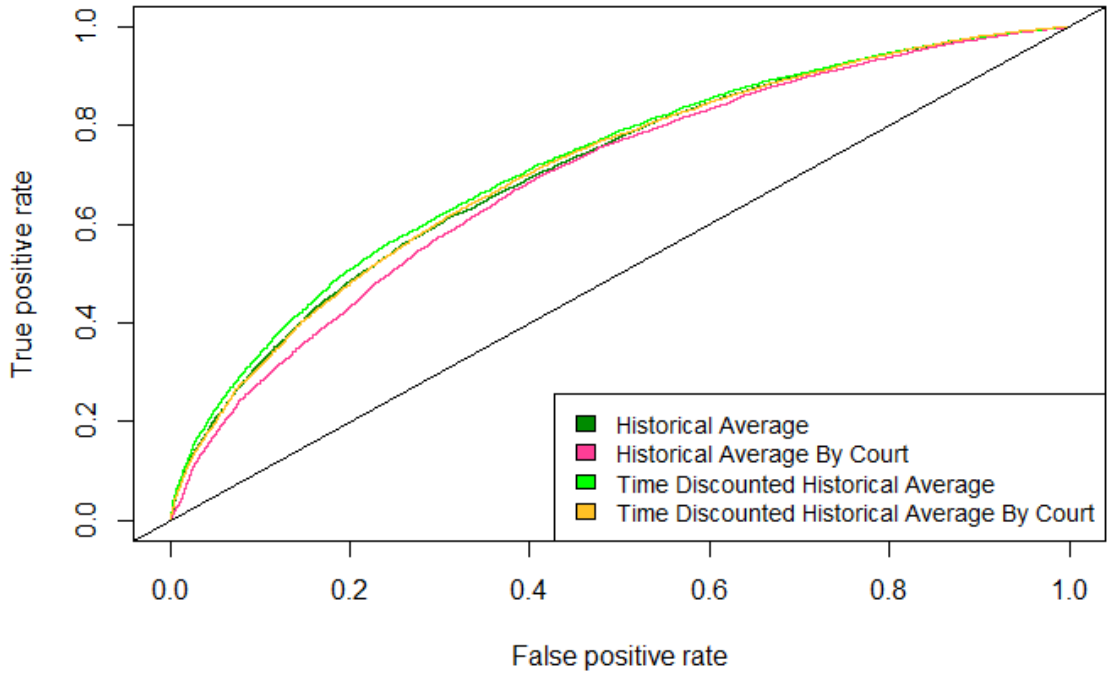


Figure 5.2: ROC curves for Engineered Features

Table 5.1: Optimised Logistic Regression Results

Feature	Historical Method	Classification Rate	Threshold
Raw	HA	60.55%	0.60
	HA BC	45.60%	0.59
	TD HA	62.58%	0.59
	TD HA BC	59.39%	0.62
Engineered	HA	60.51%	0.61
	HA BC	61.70%	0.55
	TD HA	60.62%	0.64
	TD HA BC	61.93%	0.60

HA = Historical Average
HA BC = Historical Average By Court
TD HA = Time Discounted Historical Average
TD HA BC = Time Discounted Historical Average By Court

For the optimal feature set, raw time discounted historical average, one can use forward selection to determine the best subset of features. Forward selection starts with a model only including a constant and adds one variable at a time. Variables are selected based on which decreases the Akaike information criterion (AIC), a relative measure of model goodness of fit, by the most. Only the significant variables are presented in table 5.2. The full list of variables in the final model can be found in appendix A.

Table 5.2: Final Logistic Regression Results for Raw Historical Average Feature Set

	Estimate	Std. Error	z value	Pr($>\|z\|$)
(Intercept)	0.9465	0.1370	6.91	0.0000
Player_A_weighted_rolling_average_duration	0.0123	0.0026	4.72	0.0000
Player_A_weighted_rolling_average_aces	0.0255	0.0106	2.41	0.0162
Player_A_weighted_rolling_average_double_faults	-0.0388	0.0198	-1.96	0.0496
Player_A_weighted_rolling_average_break_points_return_total	0.0883	0.0371	2.38	0.0172
Player_A_weighted_rolling_average_sets_won	1.0991	0.1986	5.53	0.0000
Player_B_weighted_rolling_average_duration	-0.0128	0.0026	-5.00	0.0000
Player_B_weighted_rolling_average_break_points_return_total	-0.0679	0.0280	-2.43	0.0152
Player_B_weighted_rolling_average_sets_won	-1.0507	0.1515	-6.94	0.0000

All the estimates are on the log-odds scale, meaning we need to exponentiate them to be able to make statements about the odds of Player A winning the match. If Player A were to increase their weighted rolling average number of aces per match by one, then the odds of Player A winning the match are expected to increase by 2.58% (i.e. multiplied by $e^{0.0255}$), holding everything else constant. Conversely, if Player A were to increase their weighted rolling average number of double faults per match by one, then the odds of Player A winning the match are expected to decrease by 3.81% (i.e. odds get multiplied by $e^{-0.0388}$) ceteris paribus.

In section 3.3.3 we mentioned that a shorter average match duration could indicate that a player is less fatigued. We also said it is possible that a shorter match duration is an indication that a player is more clinical, since they spend less time on court. However, the weighted rolling average duration for Player A and B have positive and negative coefficients respectively. Thus, a higher match duration for either increases the odds of that player winning the match. This is contrary to what we previously suggested. These coefficients could be an indication of a higher average match duration rather meaning the player is less likely to be beaten easily.

Table 5.3: Confusion Matrix for Raw Historical Average Feature Set

	Observed	
Predicted	Player A	Player B
Player A	824	363
Player B	611	805

Finally, table 5.3 shows the confusion matrix for the optimal model, raw time discounted historical average. From this confusion matrix one can calculate sensitivity and specificity values.

$$\text{Sensitivity} = \frac{824}{824+363} \approx 0.6942$$
$$\text{Specificity} = \frac{805}{805+611} \approx 0.5685$$

This means that for the 2019 dataset, 69.42% of the time we correctly predict Player A as the winner and 56.85% of the time we correctly predict Player B as the winner. The overall classification accuracy, when using a logistic regression to predict tennis match outcomes in 2019, is 62.58%.

5.1.2 Classification Trees

The initial classification tree fitting process begins with a large, unpruned tree as outlined in section 4.2. In order to do this, we use the tree package in R and the tree.control function to set the depth of the tree. Tree.control enables us to set the tree depth by changing 3 parameters, below, all of which create a deeper tree the smaller they are.

1. The minimum number of observations that must be included in both child nodes after the split is made. We set this as 2.
2. The smallest allowed node size where a split will be considered. We set this as 4.
3. The splitting criteria index of the child node must be at least this parameter times the root node splitting criteria value. We set this as 0.001.

Due to the nature of our data, these trees are not extremely large despite using small parameter values. All of the unpruned trees can be found in appendix B, for both raw and engineered feature sets.

However, these large trees often suffer from overfitting. As explained in section 4.2.4, this means they will perform poorly on unseen data. Pruning is performed using the method that is outlined in section 4.2.4 and the plots of the cost complexity cross-validation procedure can be found in figures B.9 and B.10.

These plots show that the cost complexity criteria is always minimised with either 3 or 4 terminal nodes, after which it remains the same. This indicates that the added complexity may not benefit the classification tree models.

The classification trees for the raw feature sets split either on Player A's sets won or Player A's seed. This is logical, as the more sets a player wins on average the more likely they are to win a match. The seed being an important raw feature is also logical, as this gives an indication of a player's rank at the time of the match.

Interestingly, for the engineered feature sets, cost complexity pruning results in the same tree, as seen in appendix B. For all of these classification trees the overall serve advantage feature resulted in the largest reduction in the splitting criteria. This was followed by the seeding difference. Later in this section we will interpret the best model for raw and engineered features in more detail.

Table 5.4: Optimised Classification Tree Results

Feature	Historical Method	CR Unpruned	CR Pruned	Num. Terminal Nodes
Raw	HA	60.51%	59.97%	3
	HA BC	61.20%	59.93%	4
	TD HA	61.66%	60.05%	3
	TD HA BC	62.20%	59.93%	4
Engineered	HA	61.97%	60.47%	3
	HA BC	62.16%	60.47%	3
	TD HA	63.27%	60.47%	3
	TD HA BC	62.70%	60.47%	3

CR = Classification Rate
HA = Historical Average
HA BC = Historical Average By Court
TD HA = Time Discounted Historical Average
TD HA BC = Time Discounted Historical Average By Court

Now that we have both the unpruned and pruned classification trees, we are able to investigate how they perform on the test data. The results from the classification tree models are shown in table 5.4. In this table we can see that the unpruned trees always performed better on unseen data. This shows that the large trees are not overfit, and that the increased complexity gives better results. The best feature set for the raw features is the time discounted historical average by court. Unfortunately, it is difficult to graphically interpret large trees.

As can be seen in figure 5.3, the plot becomes full and difficult to interpret. However, we can still demonstrate how to read the tree. By starting at the root node, we move to the left split if the condition is true or right if it is false. Thus, if Player A's seed is less than 28.5, we move to the child node on the left. Subsequently, if Player B's seed is less than 3.5, we would predict Player B as the winner of the tennis match. We continue this process throughout the tree, until we reach each terminal node. We can see how often the seed and average number of sets won variables are used to split on, showing that these variables are extremely important in predicting the outcome of a tennis match. Furthermore, the length of the vertical lines represent how much each split reduces the splitting criteria. We can see that the Player A seed variable creates a major reduction, as does Player B seed.

For engineered features, time discounted historical averaging is the best feature set. This is the best classification rate across all of the feature sets too. We encounter a similar problem with the interpretation, as can be seen in figure 5.4. The most notable characteristic of this figure is how drastically the overall serve advantage decreases the splitting criteria. No other feature decreases the splitting criteria by nearly the same amount, showing how important the overall serve advantage feature is when predicting tennis match outcomes.

Overall, the best classification tree was able to predict the outcome of 2019 tennis matches 63.27% of the time, using our engineered features and time discounted historical averaging.

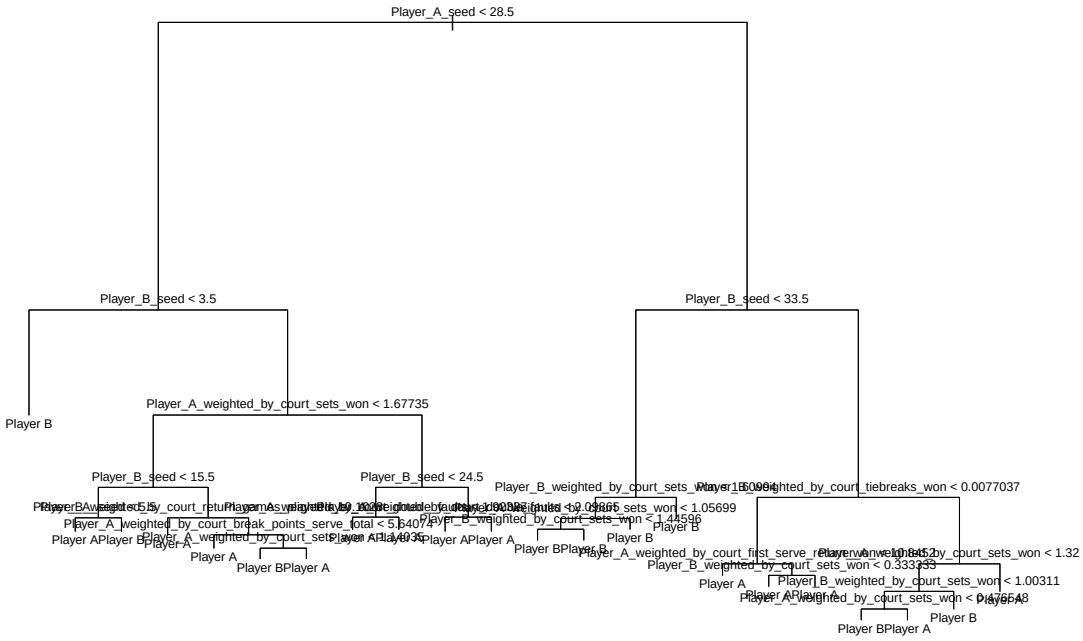


Figure 5.3: Best Classification Tree with Raw Features

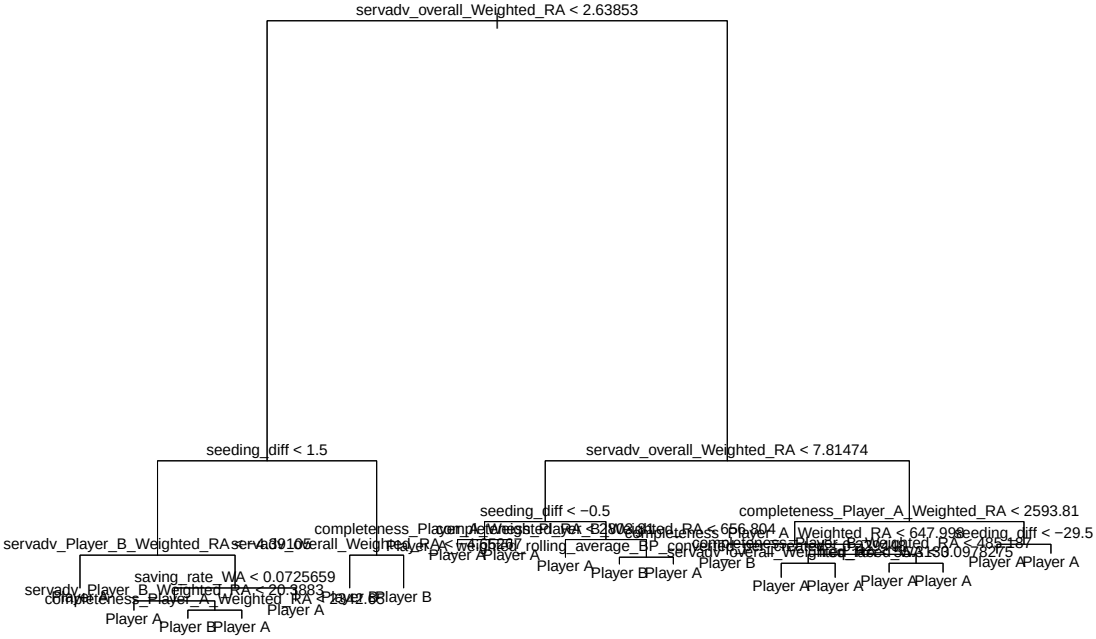


Figure 5.4: Best Classification Tree with Engineered Features

5.1.3 Random Forest

During the random forest model fitting procedure, we created two random forests per feature set. First, we fit an unoptimised random forest using the randomForest package in R and the default parameters. Following this, an optimised model was created using the caret and ranger packages. The detailed explanation of random forests is given in section 4.3 and this section will only outline the application and results.

First, we will look at the unoptimised model. This was fit to each feature set using the default parameters for classification. These are given as 500 trees, $m = \sqrt{p}$, minimum node-size = 1 and splitting criteria as the Gini index. The main purpose for fitting the unoptimised random forest models is to find the optimal number of trees. The caret package is not able to optimise this hyperparameter and it must be set. First, we fit these trees on the raw feature sets and plot the out-of-bag errors in figure 5.5. In this plot we can see that the random forest for every raw feature set appears to become stable around 300 trees, indicating that a feasible solution is found using a much smaller random forest than the default.

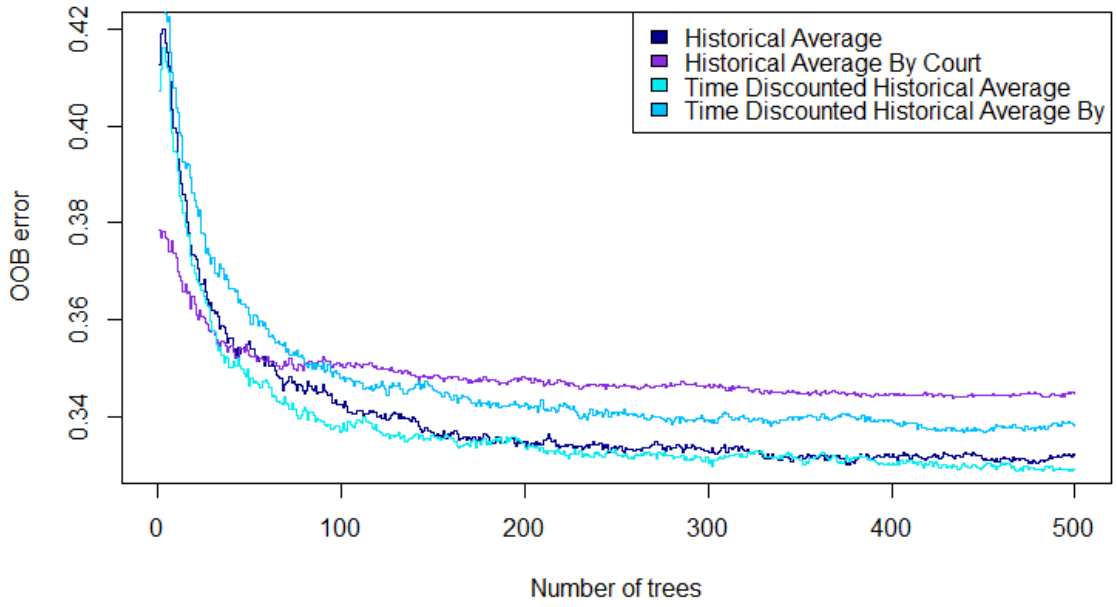


Figure 5.5: Out-of-bag Error from Random Forest Model for Raw Features

We repeat this process for the engineered features in figure 5.6 below. Again, we observe that all of the random forests have a stable out-of-bag error rate by 300 trees. For this reason we can set the optimised number of trees at 350 and be confident that all random forest models will find a feasible solution at this size. While these forests could have fewer trees, this hyperparameter value ensures that any model will find an optimal solution for our data.

Next, we can begin to fit the optimised models in R. This is done using a grid search, implemented using the caret package. The caret package is able to train and create many random forests by using the ranger implementation (Barter, 2017). The optimised random forest results are given in table 5.5.

From this table we can see that the best classification rate for raw features was found using the time discounted historical average feature set. The optimal classification rate for engineered features was achieved using the time discounted historical average by court feature

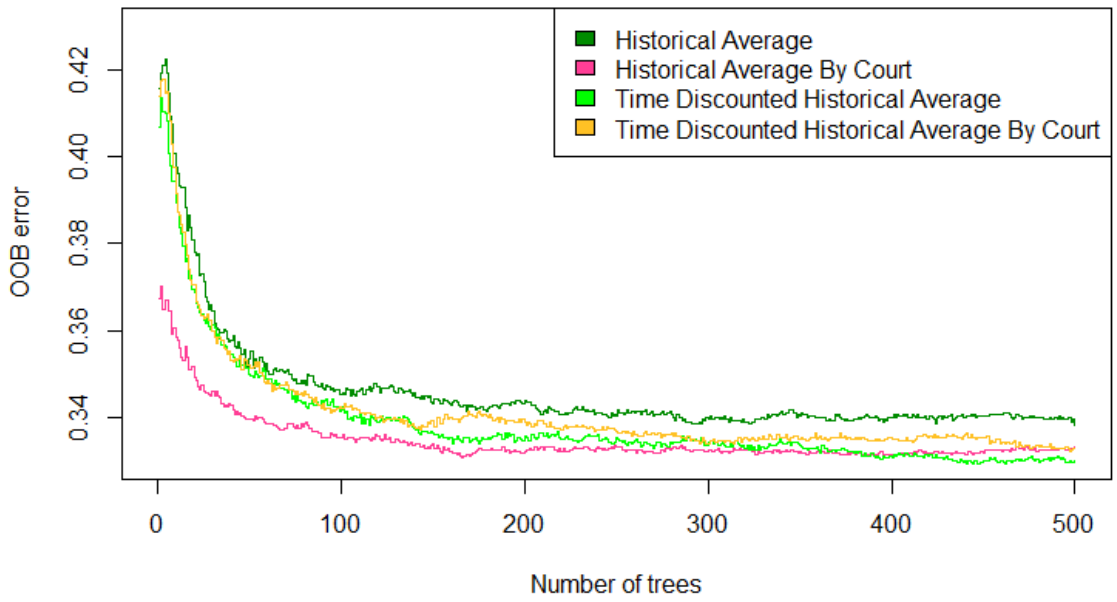


Figure 5.6: Out-of-bag Error from Random Forest Model for Engineered Features

Table 5.5: Optimised Random Forest Results

Feature	Historical Method	Classification Rate	m	Min Node Size
Raw	HA	62.16%	7	1
	HA BC	60.81%	7	5
	TD HA	62.62%	6	3
	TD HA BC	62.50%	6	1
Engineered	HA	62.27%	2	1
	HA BC	62.39%	5	3
	TD HA	62.62%	3	1
	TD HA BC	62.97%	2	3

HA = Historical Average
HA BC = Historical Average By Court
TD HA = Time Discounted Historical Average
TD HA BC = Time Discounted Historical Average By Court

set. The engineered features outperformed the raw features, although only by a slim margin. In general, the difference in classification rate across both the feature types and the historical averaging methods was small. The only feature set that performed significantly worse was the raw features that were historically averaged by court. This indicates that the engineered features did not provide a significant improvement in classification accuracy, nor did the type of historical averaging. This is similar to all models and will be discussed in section 5.2 below.

Interestingly, the optimal parameter values differed immensely across the models. This is simply due to how similar each model was, regardless of the parameters. If we investigate our best overall model, engineered features using the time discounted historical average by court, we can see that the 5 fold cross-validation results, given in appendix C, across tuning parameters are all incredibly similar. This indicates that parameters have a minimal impact on the prediction accuracy.

As discussed in section 4.3, the interpretation of a random forest model is difficult. We need to make use of variable importance plots in order to interpret the model. The variable importance plots for all the models can be found in appendix C, while this section will only discuss the optimal model.

The variable importance plot for the engineered features using the time discounted historical average by court is given in plot 5.7. In this plot we can see that the serve and return features are by far the most important when predicting tennis match outcomes. Not only was the overall serve advantage the most important, the serve advantage for Player A and Player B as well as their completeness were among the most important features. This indicates that the serve and return features do capture the dynamics of a tennis match as intended, and comparing a player's serve to their opponent's return is useful in predicting tennis match outcome.

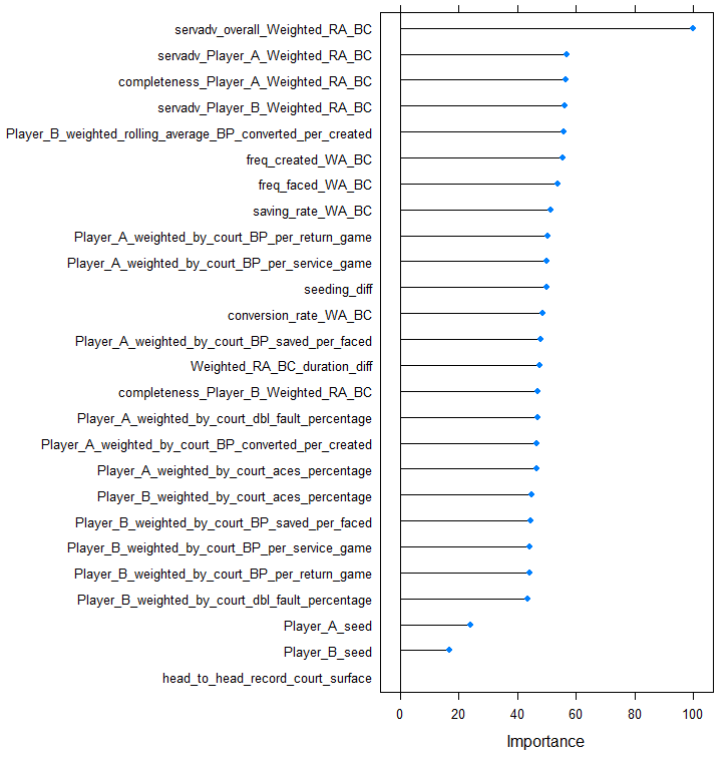


Figure 5.7: Random Forest Engineered Features Time Discounted Historical Average By Court Variable Importance

5.1.4 Gradient Boosting

As previously mentioned in section 4.4, for a gradient boosting model we need to tune the hyperparameters d and λ, as well as the number of trees. This was done in R, using the caret package. A grid search was performed for each of the eight feature sets, with the following values trialled for each parameter:

$$\begin{aligned} \text{Number of trees} &= 500, 750, 1000 \\ d &= 1, 5 \\ \lambda &= 0.1, 0.001 \\ \text{Minimum node size} &= 1 \end{aligned}$$

The classification rate on the 2019 test set, as well as the optimal combination of hyperparameters, are given for each feature set in table 5.6. As one can see all feature sets use a λ value of 0.1 and all but one feature set use one split per tree. Of the raw feature sets, the time discounted historical average and time discounted historical average by court perform best with a classification rate of 62.81%. For the engineered features, time discounted historical average performs best with a classification rate of 63.89%. Although the classification rates for engineered feature sets are higher, the improvement is only marginal.

Table 5.6: Optimised Gradient Boosting Results

Feature	Historical Method	Classification Rate	Number of Trees	d	λ	Min Node Size
Raw	HA	61.85%	500	1	0.1	1
	HA BC	57.16%	750	1	0.1	1
	TD HA	62.81%	1000	1	0.1	1
	TD HA BC	62.81%	750	1	0.1	1
Engineered	HA	61.89%	750	1	0.1	1
	HA BC	62.50%	500	5	0.1	1
	TD HA	63.89%	500	1	0.1	1
	TD HA BC	63.77%	750	1	0.1	1

HA = Historical Average
HA BC = Historical Average By Court
TD HA = Time Discounted Historical Average
TD HA BC = Time Discounted Historical Average By Court

As was done for random forests, we plot the variable importance for the best gradient boosting model, which is shown in figure 5.8. This is the engineered feature set with time discounted historical averaging. The variable importance plots of the other feature sets can be found in appendix D.

The most important feature in predicting tennis match outcomes, using a gradient boosting model, is the overall serve advantage. This makes sense given the importance of serving and returning within a tennis match. It also suggests that our serve and return feature does capture the serve-return dynamic of tennis. The second most important feature is the seeding difference. This means the seeding difference of two players in a match gives a reasonable measure of the strength of the respective players, and who is more likely to win the match. Other features do not really play any role.

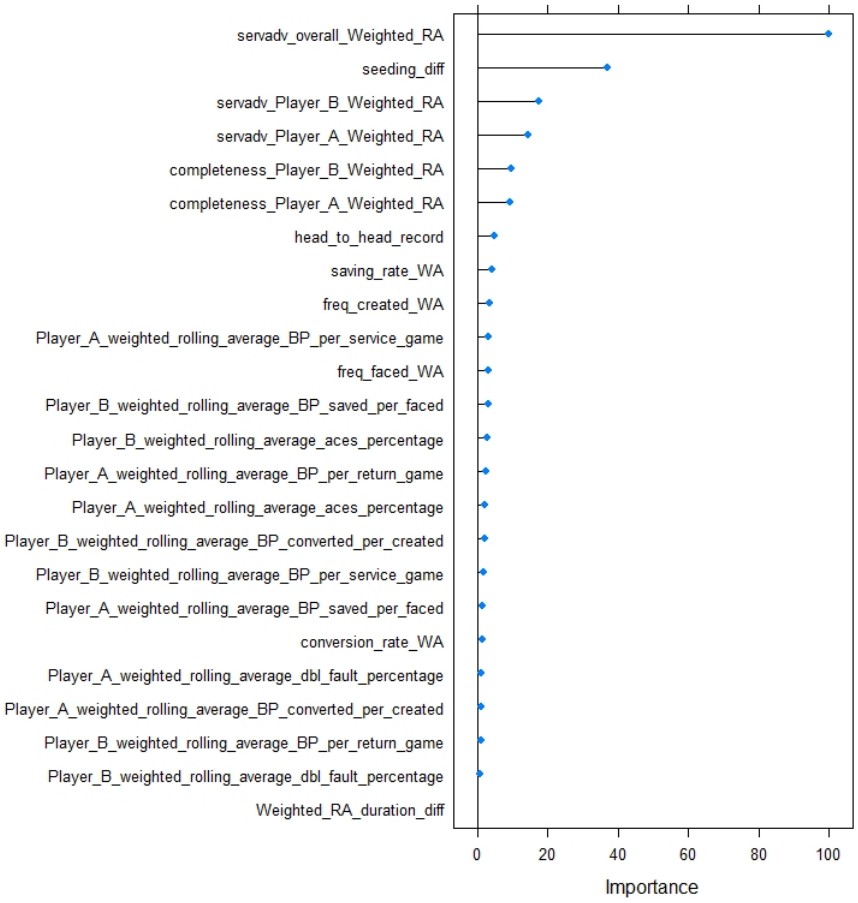


Figure 5.8: Gradient Boosting Engineered Features Time Discounted Historical Average Variable Importance

5.2 Model Comparison

After fitting four different supervised learning models to eight different feature sets, we can now compare how each model performed when predicting tennis match outcomes. Furthermore, we can investigate whether the features that we engineered predicted more accurately than just using the raw features, as well as which historical averaging method was best. First, we will create a table which shows the classification rate for each model, for every feature set. This can be found in table 5.7

Within table 5.7 each row is a model and each column is a different feature set. Highlighted in orange is the best model for each feature set, or the best classification rate per column. Highlighted in blue is the best feature set for each model, or the best classification rate per row. Finally, the cell highlighted in purple is both the best model for the feature set as well as the best feature set for the model.

From table 5.7 we can see that the gradient boosting model generally outperforms the others. For five out of the eight feature sets this model gave the highest classification rate. Time discounted historical averaging outperformed historical averaging that did not include a recency bias. Stratifying time discounted historical averaging by court was only beneficial for the random forest.

Table 5.7: Model Comparison for all Feature Sets

	Raw Features				Engineered Features			
Model	HA	HA BC	TD HA	TD HA BC	HA	HA BC	TD HA	TD HA BC
Logistic Regression	60.55%	45.60%	62.58%	59.39%	60.51%	61.70%	60.62%	61.93%
Classification Tree	60.51%	61.20%	61.66%	62.20%	61.97%	62.16%	63.27%	62.70%
Random Forest	62.16%	60.81%	62.62%	62.50%	62.27%	62.39%	62.62%	62.97%
Gradient Boosting	61.85%	57.16%	62.81%	62.81%	61.89%	62.50%	63.89%	63.77%

HA = Historical Average
HA BC = Historical Average By Court
TD HA = Time Discounted Historical Average
TD HA BC = Time Discounted Historical Average By Court

Best Model for Feature Set
Best Feature Set for Model
Best for Feature Set & Model

The best overall model was gradient boosting using the engineered time discounted historical averaging feature set. This resulted in a classification rate of 63.89% when predicting the outcome of tennis matches played in 2019. The serve advantage feature was the best engineered feature, shown by being the most important variable for both the random forest and gradient boosting models. Seeding was also an important feature, both as a raw feature and when engineered as a seeding difference. This gives a general indication of a player's skill level, which is vital when predicting tennis match outcomes.

However, something that is clear is that all of the classification rates were incredibly similar. While some classification rates were particularly poor, in general there is no significant difference between the models and feature sets. The historical average by court for raw features performed far worse than other feature sets for all of the models. No model was particularly worse than the others, however logistic regression was never the best model for a feature set.

Overall, this shows that it is difficult to predict tennis matches. While the reason for this is not entirely clear, a possible reason could be the slim margin between professional tennis players. Every player on the ATP tour is an excellent athlete, and the difference between the top ranked players is minimal. This makes it incredibly difficult for supervised learning models to distinguish the winner of a tennis match. Additionally, supervised learning techniques are all powerful tools and one is not better than the others. While some may work better in certain cases, for this application all models are equally strong predictors.

5.3 Chapter Summary

All four supervised learning algorithms gave a similar classification accuracy for the eight feature sets. Gradient boosting was the best model in general, with the best classification accuracy for five out of eight feature sets. Our optimal model was found using gradient boosting and time discounted historical averaging, with a classification rate of 63.89%. The overall serve advantage feature did out perform all other engineered features, as it was by far the most important variable for both the random forest and gradient boosting models. Seeding was an important feature too, as it gives a general indication of a players' strength.

The fact that all feature sets had similar classification rates indicates that the engineered features did not manage to model the dynamics of a tennis match better than raw features. While engineered features did generally have a slightly higher classification rate, it was not improved by a significant margin.

Chapter 6

Conclusion

This project aimed to predict tennis match outcomes by using various engineered features to better capture the dynamics of a tennis match. Only male singles tennis matches on the ATP tour were considered, with the focus being to create predictions before matches were played. Predictions were made using various supervised learning algorithms, namely logistic regression, classification trees, random forests and gradient boosting. Data had to be averaged in order to predict matches before they were played, and four different methods for creating the historical record were used.

6.1 Key Findings

Despite much consideration, the engineered features were unable to improve the classification rate of the various models. While these features did result in a higher classification rate in most cases, the difference was insignificant. Introducing a recency bias was helpful when creating an historical record; however, again the improvement in classification rate among the various methods was insignificant. Furthermore, there was minimal difference between the predictive accuracy of the models. Gradient boosting did perform best on five out of the eight feature sets, and was the supervised learning model that gave the best classification rate overall. This classification rate was 63.89%, achieved when using time discounted historical averaging for the engineered features.

Of the features that were engineered, the serve and return feature best modelled the dynamics of a tennis match. This feature manages to capture how the strength of a player's serve compares to the strength of their opponents return, and was important in model creation. Seeding was also important. Due to the fact that seeding reflects a player's rank, this also gives a good indication of skill level. During model creation, seeding was often used as a key predictor, as a raw feature as well as when differenced between the two players in a tennis match.

6.2 Future Work and Recommendations

There are many areas in which this project could be improved and extended upon. Further research could also be conducted with regards to both feature engineering and supervised learning models. Firstly, there are some match statistics which we were unable to access that may improve the prediction accuracy of the models. Winners and unforced errors are important statistics in tennis frequently used by analysts. However, data regarding these is not widely available.

Other data that is also not widely available is Hawk-Eye data. This data describes almost every individual shot hit on the ATP tour in recent years. This would let us examine more aspects of the serve and return dynamic such as location, ball speed and angle. This

data could be an excellent predictor of tennis match outcomes; however, is unavailable due to commercial rights.

This project also focused purely on men's tennis due to time constraints. We believe an analysis on women's tennis would be equally as interesting, especially as the importance of features may not be the same.

Any future work could also expand on the supervised learning techniques used. The use of neural networks, support vector machines and other regression based models is common in literature. This project could undoubtedly be extended to compare these models to the ones used.

Finally, an extension to this project that would be extremely interesting is to create a live tennis match outcome predictor. While this project focused purely on historical statistics, to combine this with the use of live tennis data would make for interesting research.

Overall, while the results that we hoped for were not achieved, the process was still valuable. The theory behind the features is sound despite a low classification accuracy. While it is difficult to compare to past research as few papers publish their test classification rate, our predictions were still better than chance. There are also many areas where this paper could be extended, and much scope for continued research in the field of tennis.

Bibliography

Amaratunga, Dhammika, Javier Cabrera, and Yung-Seop Lee (2008). "Enriched random forests". In: *Bioinformatics* 24.18, pp. 2010–2014.

ATP Tour (2020). *ATP Website*. URL: https://www.atptour.com/.

Barnett, Tristan and Stephen R Clarke (2005). "Combining player statistics to predict outcomes of tennis matches". In: *IMA Journal of Management Mathematics* 16.2, pp. 113–120.

Barter, Rebecca (2017). *A basic tutorial of caret: the machine learning package in R*. URL: http://www.rebeccabarter.com/blog/2017-11-17-caret_tutorial/.

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Britz, Stefan and Miguel Lacerda (2020a). "Statistical Sciences Honours Analytics 2020, Bagging & Random Forests". University Lecture.

– (2020b). "Statistical Sciences Honours Analytics 2020, Boosting". University Lecture.

– (2020c). "Statistical Sciences Honours Analytics 2020, Classification Trees". University Lecture.

Brownlee, Jason (2016). *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning*. URL: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

Cornman, Andre, Grant Spellman, and Daniel Wright (2017). *Machine learning for professional tennis match prediction and betting*. Tech. rep. Working Paper, Stanford University, December.

Cramer, Jan Salomon (2002). "The origins of logistic regression". In:

Davenport, Thomas H (2014). "Analytics in sports: The new science of winning". In: *International Institute for Analytics* 2, pp. 1–28.

Devcic, John (2020). *Weighted Moving Averages: The Basics*. URL: https://www.investopedia.com/articles/technical/060401.asp.

Eryarsoy, Enes and Dursun Delen (2019). "Predicting the Outcome of a Football Game: A Comparative Analysis of Single and Ensemble Analytics Methods". In: *Proceedings of the 52nd Hawaii International Conference on System Sciences*.

Fawagreh, Khaled, Mohamed Medhat Gaber, and Eyad Elyan (2014). "Random forests: from early developments to recent advancements". In: *Systems Science & Control Engineering: An Open Access Journal* 2.1, pp. 602–609.

Friedman, Jerome H (2001). "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics*, pp. 1189–1232.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York.

Glass, Harry (2012). *Betfair makes a Murray mint as US Open and Wimbledon finals see punters place record bets*. URL: https://www.thisismoney.co.uk/money/markets/article-2201610/Betfair-makes-Murray-mint-US-Open-Wimbledon-finals-punters-place-record-bets.html.

Gruetzemacher, Ross, Ashish Gupta, and Gary B Wilkerson (2016). "Sports Injury Prevention Screen (SIPS): Design and Architecture of an Internet of Things (IoT) Based Analytics Health App." In: *CONF-IRM*, p. 18.

Hastie, Trevor (2003). "Boosting". University Lecture.

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

Judd, Dan, Rebecca Wu, and Camillo Jose Taylor (2014). "Squash sport analytics & image processing". In: MIT Sloan Sports Analytics Conference.

Katić, Ratko et al. (2011). "Impact of game elements on tennis match outcome in Wimbledon and Roland Garros 2009". In: *Collegium antropologicum* 35.2, pp. 341–346.

Kovalchik, Stephanie Ann (2016). "Searching for the GOAT of tennis win prediction". In: *Journal of Quantitative Analysis in Sports* 12.3, pp. 127–138.

Leung, Carson K and Kyle W Joseph (2014). "Sports data mining: predicting results for the college football games". In: *Procedia Computer Science* 35, pp. 710–719.

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

Lin, Kevin and Andrew Juko (2020). *Serve and Volley*. URL: https://github.com/serve-and-volley/atp-world-tour-tennis-data#part-c.

Loh, Wei-Yin (2014). "Fifty Years of Classification and Regression Trees 1". In:

Ma, Shang-Min et al. (2013). "Winning matches in Grand Slam men's singles: An analysis of player performance-related variables from 1991 to 2008". In: *Journal of sports sciences* 31.11, pp. 1147–1155.

McGarry, James T (1993). "The development of a stochastic model for predicting championship squash performance". PhD thesis. University of British Columbia.

Pathak, Neeraj and Hardik Wadhwa (2016). "Applications of modern classification techniques to predict the outcome of ODI cricket". In: *Procedia Computer Science* 87, pp. 55–60.

Sarlis, Vangelis and Christos Tjortjis (2020). "Sports analytics–Evaluation of basketball players and team performance". In: *Information Systems*, p. 101562.

Sipko, Michal and William Knottenbelt (2015). "Machine learning for the prediction of professional tennis matches". In: *MEng computing-final year project, Imperial College London*.

Sisense (2020). *Data Cleaning*. URL: https://www.sisense.com/glossary/data-cleaning/.

StatQuest (2019). *Gradient Boost Part 3: Classification*. Youtube. URL: https://www.youtube.com/watch?v=jxuNLH5dXCs.

The Video Analyst (2020). *The History of Sports Analysis*. URL: https://thevideoanalyst.com/history-sports-analysis/.

Tsymbal, Alexey, Mykola Pechenizkiy, and Pádraig Cunningham (2006). "Dynamic integration with random forests". In: *European conference on machine learning*. Springer, pp. 801–808.

Valero, C Soto (2016). "Predicting Win-Loss outcomes in MLB regular season games–A comparative study using data mining methods". In: *International Journal of Computer Science in Sport* 15.2, pp. 91–112.

Wang, Jiachen et al. (2019). "Tac-Simur: Tactic-based simulative visual analytics of table tennis". In: *IEEE transactions on visualization and computer graphics* 26.1, pp. 407–417.

Wilkens, Sascha (2020). "Sports Prediction and Betting Models in the Machine Learning Age: The Case of Tennis". In: *Available at SSRN 3506302*.

World Atlas (2020). *The Most Popular Sports In The World*. URL: https://www.worldatlas.com/articles/what-are-the-most-popular-sports-in-the-world.html/.

Ye, Shuainan et al. (2020). "Shuttlespace: Exploring and analyzing movement trajectory in immersive visualization". In: *IEEE Transactions on Visualization and Computer Graphics*.

Yiu, Tony (2019). *Understanding Random Forest: How the Algorithm Works and Why it Is So Effective*. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Yogonet (2020). *Global sports betting market expected to reach nearly $155.49 B by 2024*. URL: https://www.yogonet.com/international/noticias/2020/06/24/53713-global-sports-betting-market-expected-to-reach-nearly-15549-b-by-2024.

Appendix A

Logistic Regression

Table A.1: Final Logistic Regression Results

	Estimate	Std. Error	z value	Pr(> |z|)
(Intercept)	0.9465	0.1370	6.91	0.0000
Player_A_weighted_rolling_average_duration	0.0123	0.0026	4.72	0.0000
Player_A_weighted_rolling_average_aces	0.0255	0.0106	2.41	0.0162
Player_A_weighted_rolling_average_double_faults	-0.0388	0.0198	-1.96	0.0496
Player_A_weighted_rolling_average_first_serves_in	-1433.7012	2739.8953	-0.52	0.6008
Player_A_weighted_rolling_average_first_serves_total	1209.1592	2755.4261	0.44	0.6608
Player_A_weighted_rolling_average_first_serve_points_won	-1885.3504	3161.2365	-0.60	0.5509
Player_A_weighted_rolling_average_second_serve_points_won	-1885.3160	3161.2366	-0.60	0.5509
Player_A_weighted_rolling_average_second_serve_points_total	-1433.7153	2739.8954	-0.52	0.6008
Player_A_weighted_rolling_average_break_points_saved	-0.0308	0.3443	-0.09	0.9288
Player_A_weighted_rolling_average_break_points_serve_total	-0.0440	0.3403	-0.13	0.8971
Player_A_weighted_rolling_average_service_games_played	0.1570	0.3450	0.46	0.6490
Player_A_weighted_rolling_average_first_serve_return_won	-4541.5901	3182.3614	-1.43	0.1535
Player_A_weighted_rolling_average_first_serve_return_total	-247.0055	2821.3688	-0.09	0.9302
Player_A_weighted_rolling_average_second_serve_return_won	-4541.6062	3182.3612	-1.43	0.1535
Player_A_weighted_rolling_average_second_serve_return_total	-247.0227	2821.3689	-0.09	0.9302
Player_A_weighted_rolling_average_break_points_converted	-0.0753	0.3433	-0.22	0.8264
Player_A_weighted_rolling_average_break_points_return_total	0.0883	0.0371	2.38	0.0172
Player_A_weighted_rolling_average_return_games_played	-0.0129	0.1303	-0.10	0.9213
Player_A_weighted_rolling_average_service_points_won	586.0845	3634.4839	0.16	0.8719
Player_A_weighted_rolling_average_return_points_won	3242.3394	3635.9736	0.89	0.3725
Player_A_weighted_rolling_average_return_points_total	22.4534	2828.6139	0.01	0.9937
Player_A_weighted_rolling_average_total_points_won	1299.3796	3150.4987	0.41	0.6800
Player_A_weighted_rolling_average_total_points_total	224.4793	514.4606	0.44	0.6626
Player_A_weighted_rolling_average_sets_won	1.0991	0.1986	5.53	0.0000
Player_A_weighted_rolling_average_games_won	-0.1170	0.3214	-0.36	0.7158
Player_A_weighted_rolling_average_tiebreaks_won	-0.4096	0.3878	-1.06	0.2909
Player_B_weighted_rolling_average_duration	-0.0128	0.0026	-5.00	0.0000
Player_B_weighted_rolling_average_aces	-0.0149	0.0099	-1.51	0.1316
Player_B_weighted_rolling_average_double_faults	0.0038	0.0180	0.21	0.8330
Player_B_weighted_rolling_average_first_serves_in	705.9119	2568.8496	0.27	0.7835
Player_B_weighted_rolling_average_first_serves_total	-344.2079	2561.8309	-0.13	0.8931
Player_B_weighted_rolling_average_first_serve_points_won	-5959.9578	3235.8716	-1.84	0.0655
Player_B_weighted_rolling_average_second_serve_points_won	-5959.9352	3235.8717	-1.84	0.0655
Player_B_weighted_rolling_average_second_serve_points_total	705.9005	2568.8495	0.27	0.7835
Player_B_weighted_rolling_average_break_points_saved	0.0744	0.2894	0.26	0.7971
Player_B_weighted_rolling_average_break_points_serve_total	-0.0575	0.2872	-0.20	0.8414
Player_B_weighted_rolling_average_service_games_played	-0.2880	0.2934	-0.98	0.3263
Player_B_weighted_rolling_average_first_serve_return_won	1039.9876	3300.2041	0.32	0.7527
Player_B_weighted_rolling_average_first_serve_return_total	1896.4412	2655.6987	0.71	0.4752
Player_B_weighted_rolling_average_second_serve_return_won	1039.9392	3300.2040	0.32	0.7527
Player_B_weighted_rolling_average_second_serve_return_total	1896.4619	2655.6985	0.71	0.4752
Player_B_weighted_rolling_average_break_points_converted	-0.1856	0.2918	-0.64	0.5248
Player_B_weighted_rolling_average_break_points_return_total	-0.0679	0.0280	-2.43	0.0152
Player_B_weighted_rolling_average_return_games_played	-0.0550	0.1048	-0.52	0.5996
Player_B_weighted_rolling_average_service_points_won	3506.5211	3614.6171	0.97	0.3320
Player_B_weighted_rolling_average_return_points_won	-3493.2689	3652.8344	-0.96	0.3389
Player_B_weighted_rolling_average_return_points_total	-1534.8229	2651.9339	-0.58	0.5628
Player_B_weighted_rolling_average_total_points_won	2453.2290	2966.6733	0.83	0.4083
Player_B_weighted_rolling_average_total_points_total	-361.5721	531.2772	-0.68	0.4961
Player_B_weighted_rolling_average_sets_won	-1.0507	0.1515	-6.94	0.0000
Player_B_weighted_rolling_average_games_won	0.3188	0.2754	1.16	0.2469
Player_B_weighted_rolling_average_tiebreaks_won	0.3999	0.3174	1.26	0.2076

Appendix B

Classification Trees

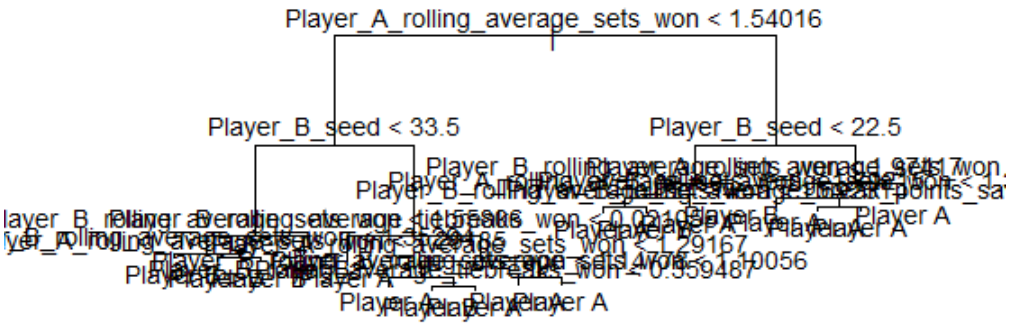


Figure B.1: Raw Historical Average Unpruned Tree

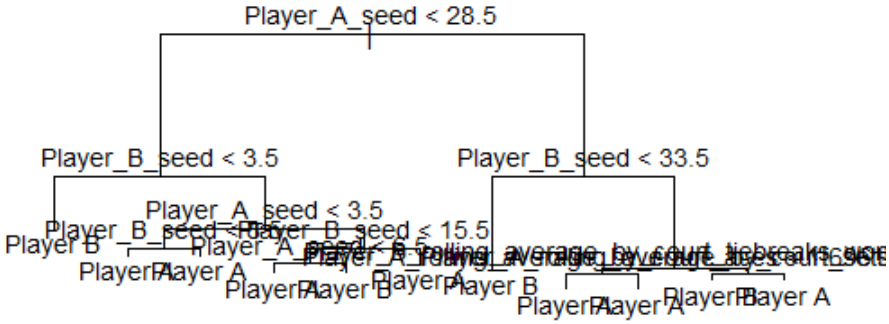


Figure B.2: Raw Historical Average By Court Unpruned Tree

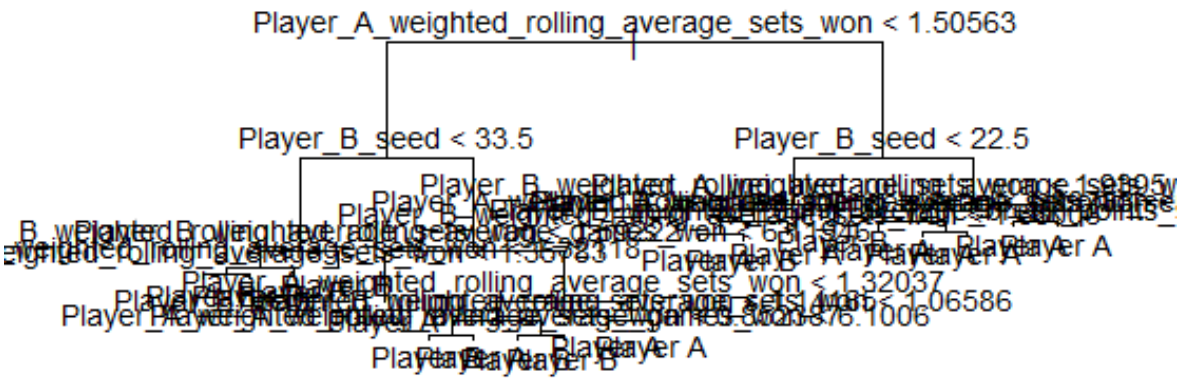


Figure B.3: Raw Time Discounted Historical Average Unpruned Tree

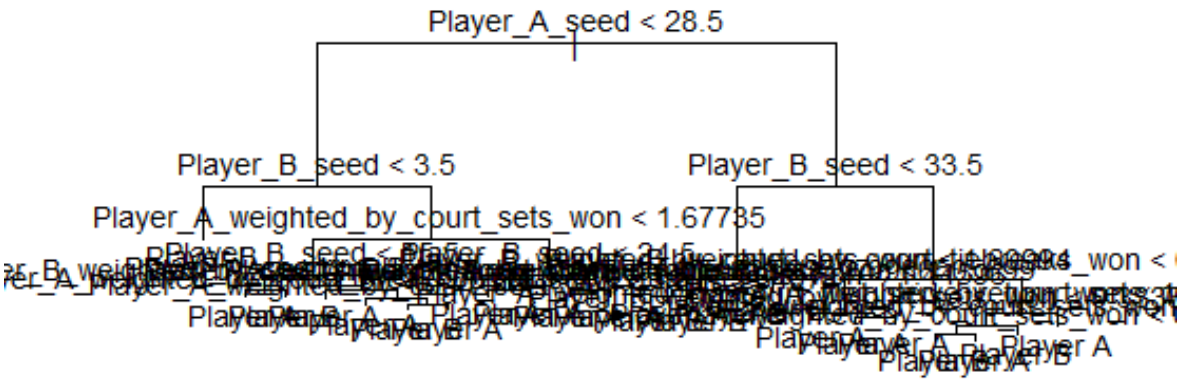


Figure B.4: Raw Time Discounted Historical Average By Court Unpruned Tree

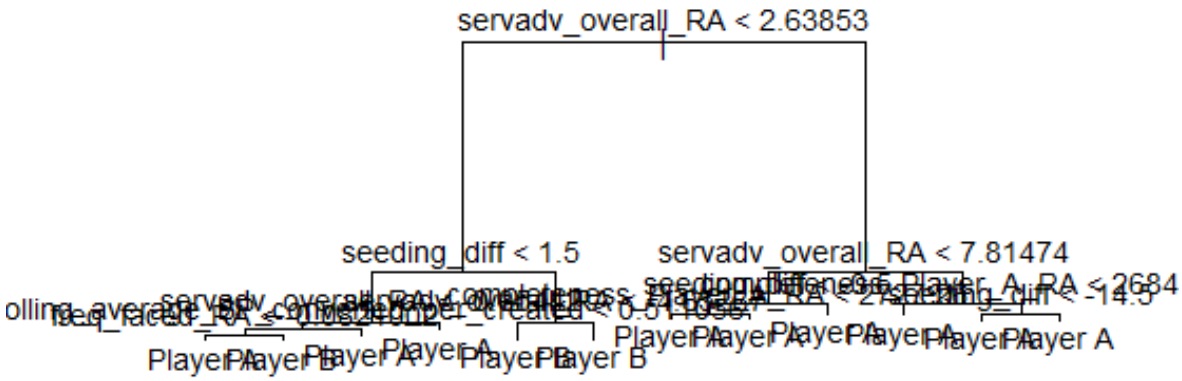


Figure B.5: Engineered Historical Average Unpruned Tree

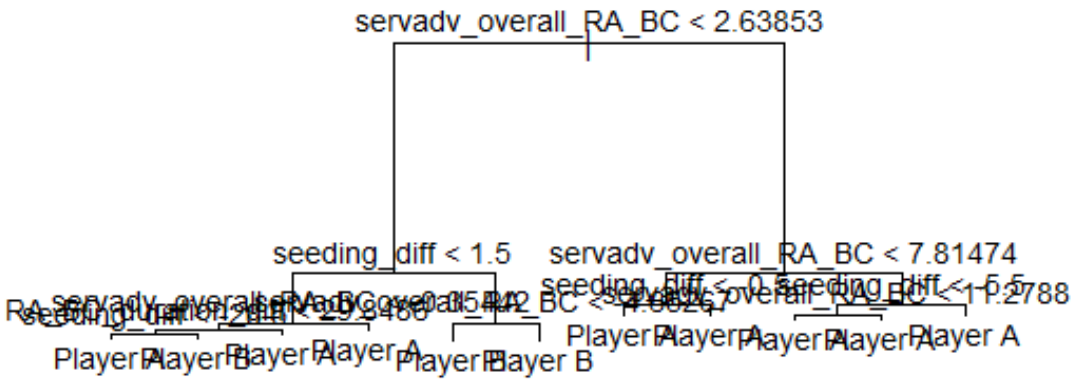


Figure B.6: Engineered Historical Average By Court Unpruned Tree

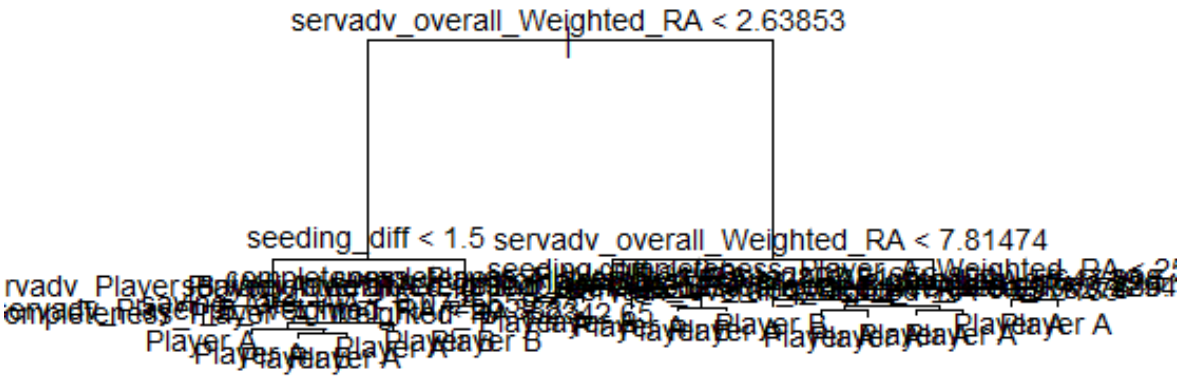


Figure B.7: Engineered Time Discounted Historical Average Unpruned Tree

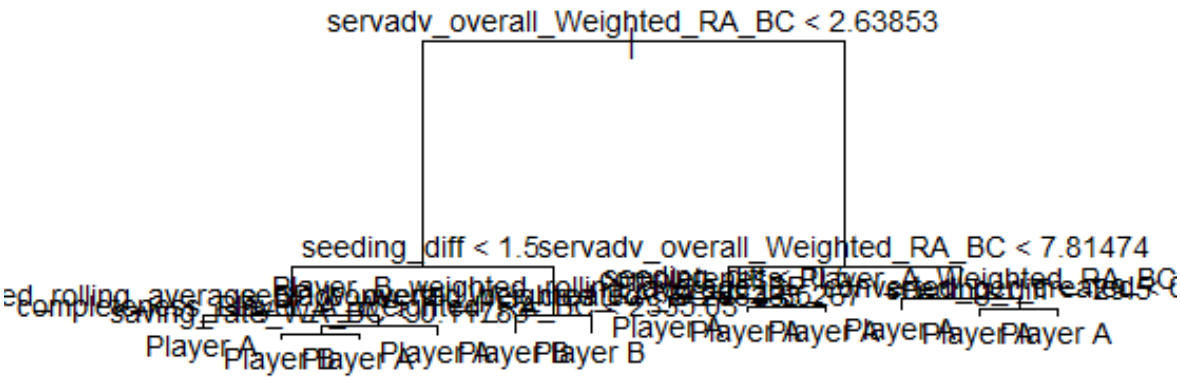
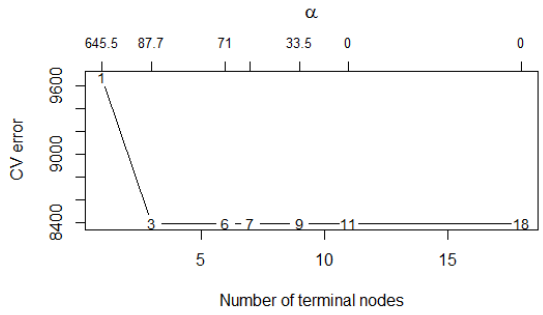
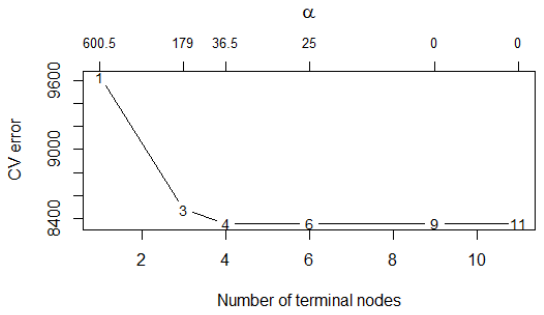


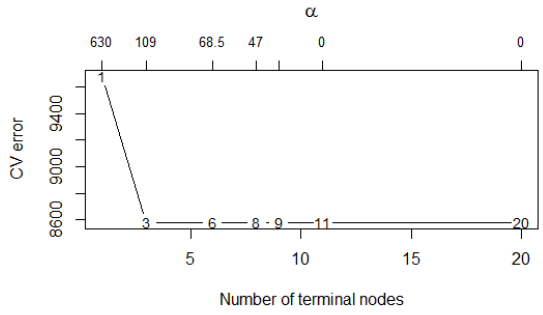
Figure B.8: Engineered Time Discounted Historical Average By Court Unpruned Tree



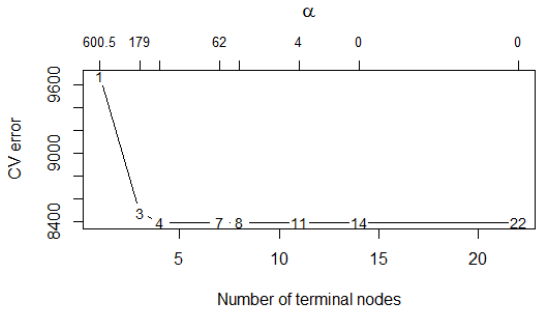
(a) Raw Historical Average



(b) Raw Historical Average

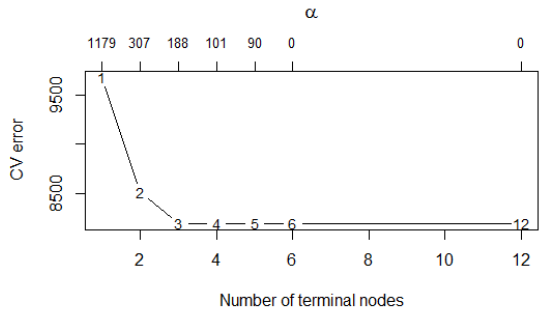


(c) Raw Time Discounted Historical Average

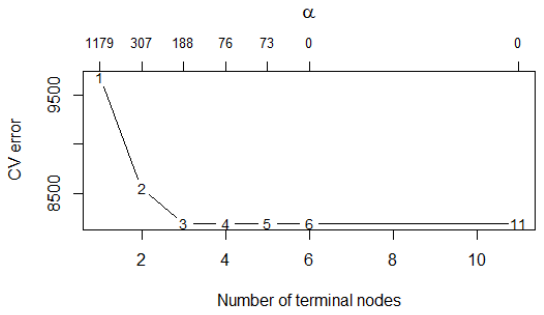


(d) Raw Time Discounted Historical Average By Court

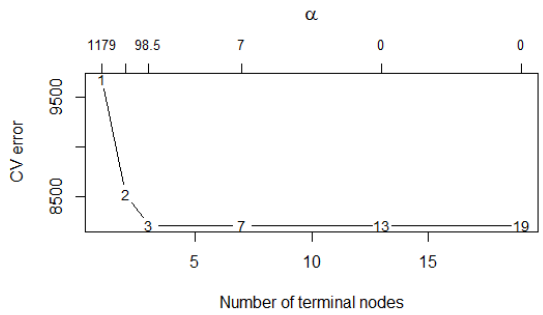
Figure B.9: Cost Complexity Cross-validation for Raw Feature Sets



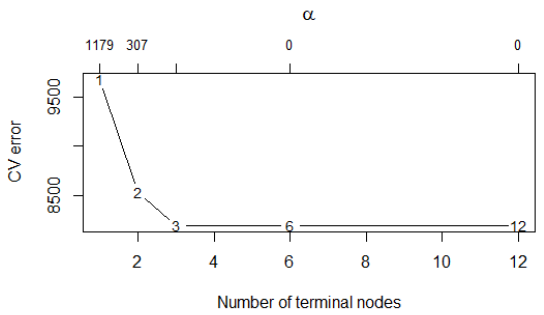
(a) Engineered Historical Average



(b) Engineered Historical Average



(c) Engineered Time Discounted Historical Average



(d) Engineered Time Discounted Historical Average By Court

Figure B.10: Cost Complexity Cross-validation for Engineered Feature Sets

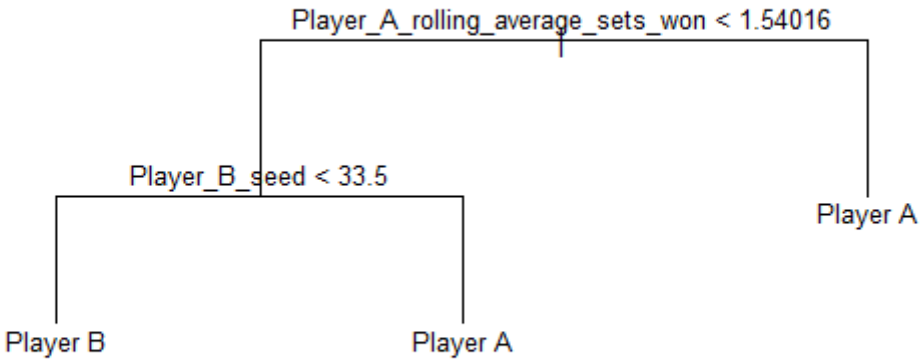


Figure B.11: Raw Historical Average Pruned Tree

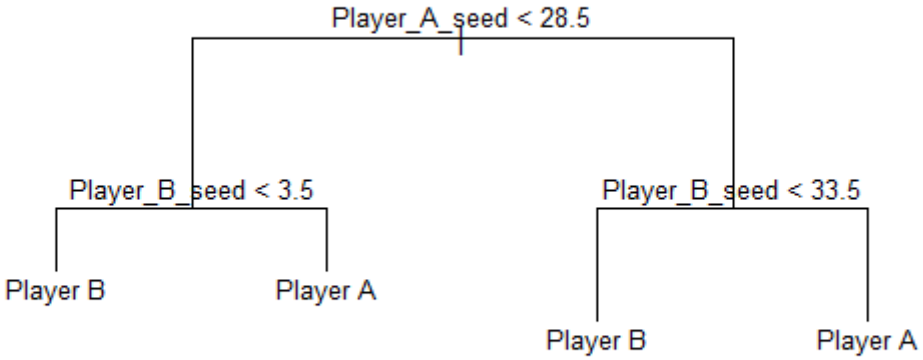


Figure B.12: Raw Historical Average By Court Pruned Tree

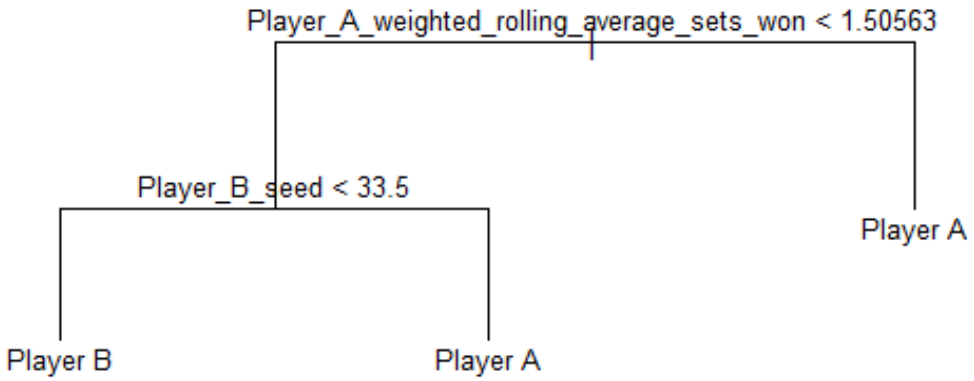


Figure B.13: Raw Time Discounted Historical Average Pruned Tree

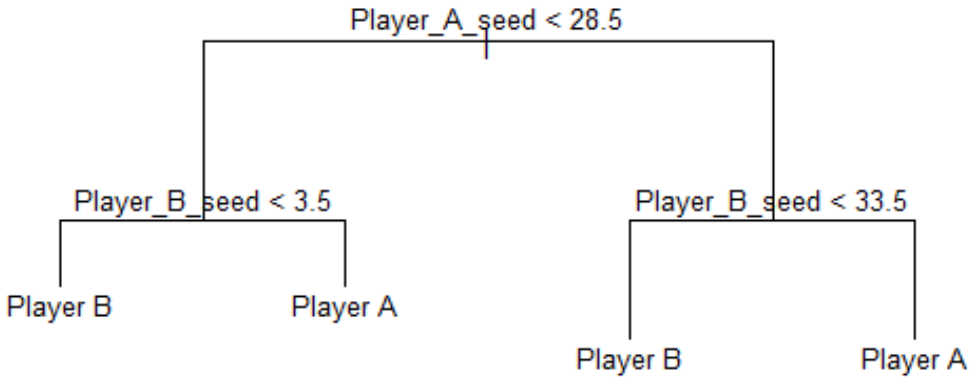


Figure B.14: Raw Time Discounted Historical Average By Court Pruned Tree

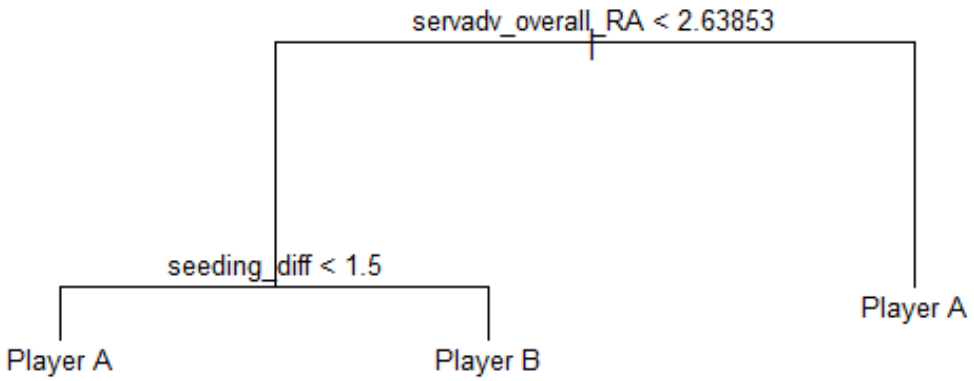


Figure B.15: Engineered Historical Average Pruned Tree

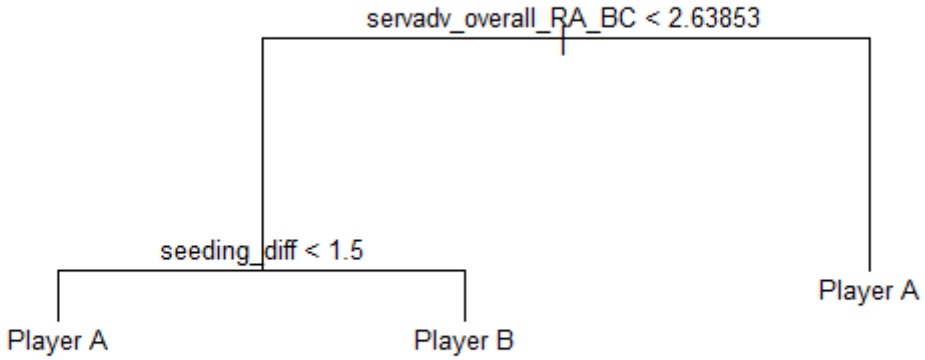


Figure B.16: Engineered Historical Average By Court Pruned Tree

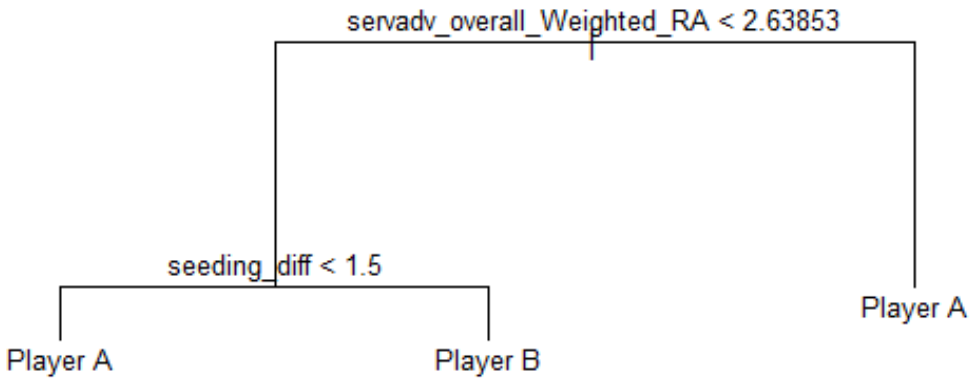


Figure B.17: Engineered Time Discounted Historical Average Pruned Tree

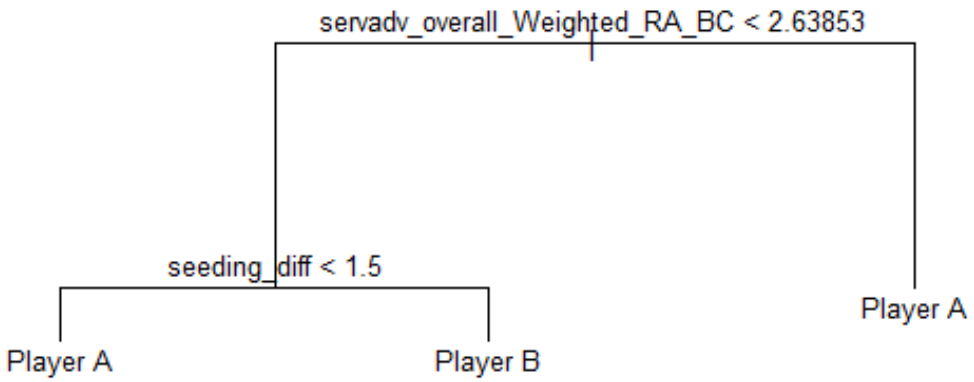


Figure B.18: Engineered Time Discounted Historical Average By Court Pruned Tree

Appendix C

Random Forest

Table C.1: 5 Fold Cross-validation Results Across Tuning Parameters

mtry	min.node.size	Accuracy	Kappa
2	1	0.6704008	0.2919674
2	3	0.6722183	0.2959536
2	5	0.6709079	0.2937863
2	10	0.6697244	0.2900872
3	1	0.6688789	0.2901759
3	3	0.6673571	0.2862816
3	5	0.6702315	0.2922458
3	10	0.6700625	0.2913865
4	1	0.6708233	0.2947323
4	3	0.6679914	0.2883801
4	5	0.6688365	0.2898877
4	10	0.6691748	0.2901519
5	1	0.6678644	0.2886355
5	3	0.6684985	0.2899146
5	5	0.6699357	0.2929790
5	10	0.6683295	0.2890329
6	1	0.6663004	0.2858657
6	3	0.6678222	0.2883968
6	5	0.6678645	0.2887059
6	10	0.6677376	0.2875093
7	1	0.6649900	0.2831023
7	3	0.6658778	0.2847289
7	5	0.6660044	0.2852673
7	10	0.6661315	0.2852358

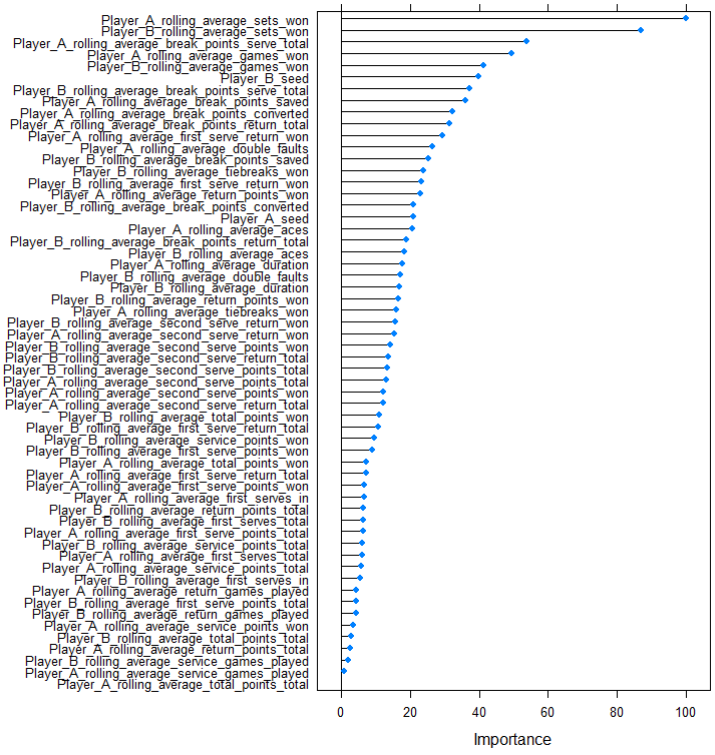


Figure C.1: Raw Features Historical Average

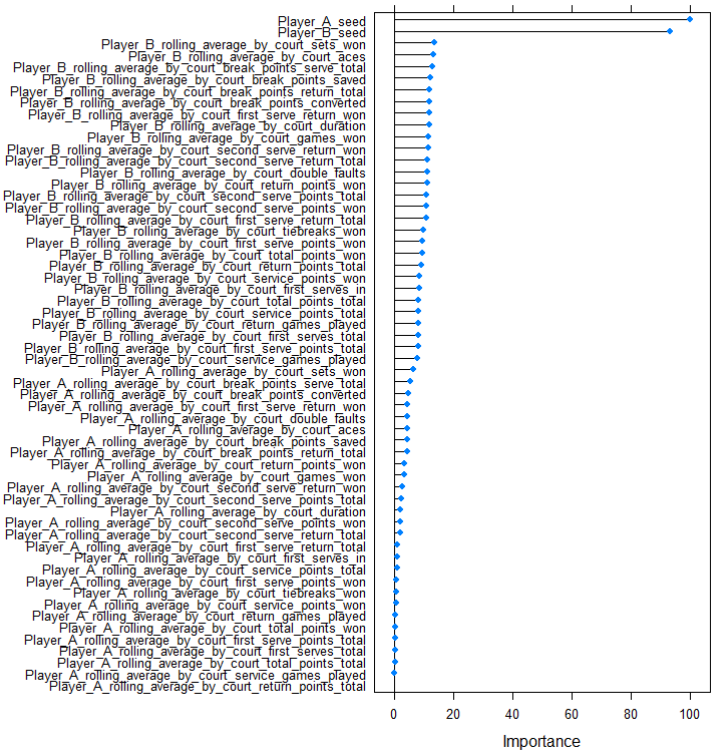


Figure C.2: Raw Features Historical Average By Court

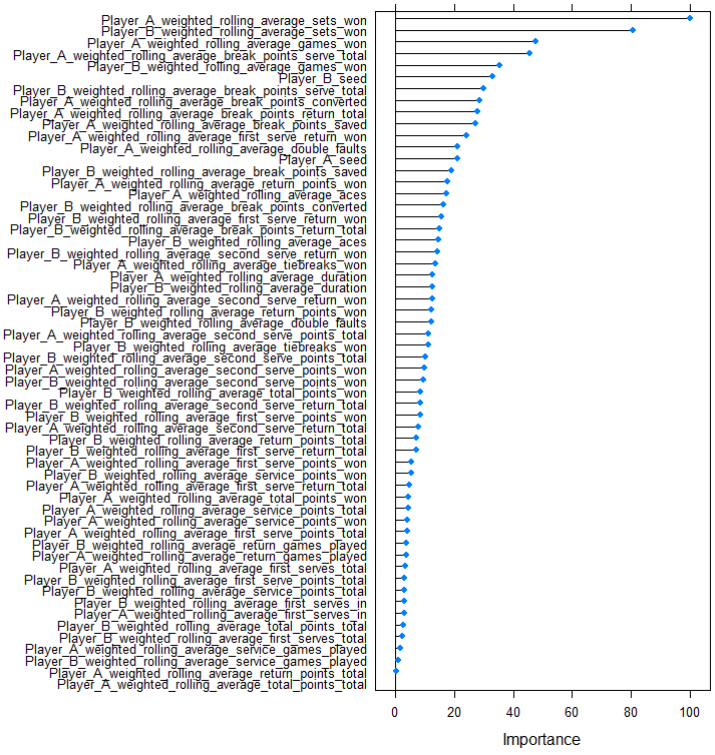


Figure C.3: Raw Features Time Discounted Historical Average

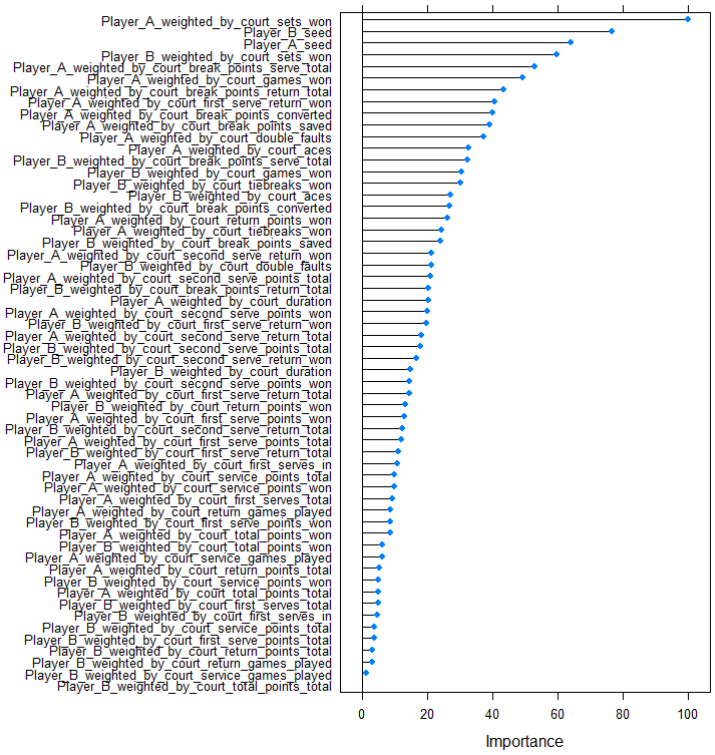


Figure C.4: Raw Features Time Discounted Historical Average By Court

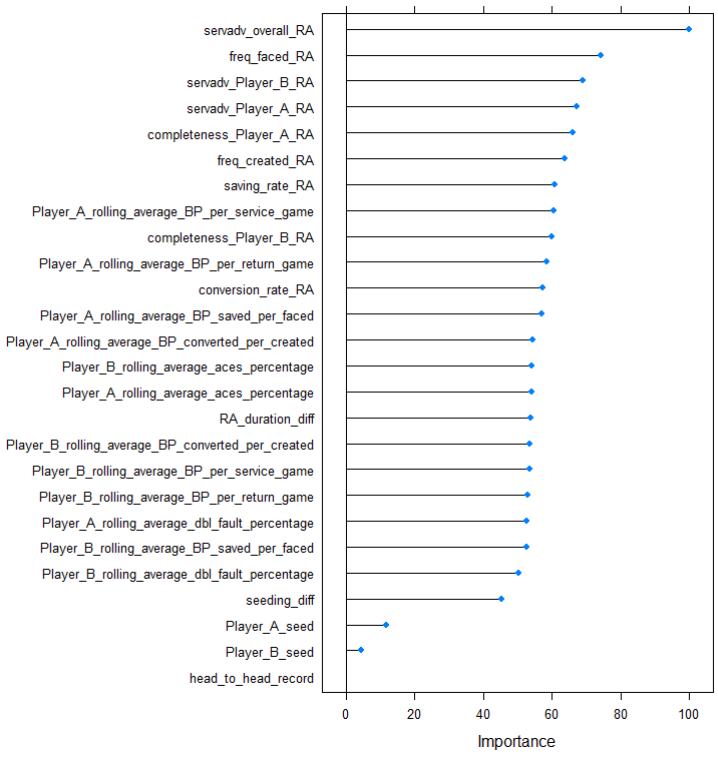


Figure C.5: Engineered Features Historical Average

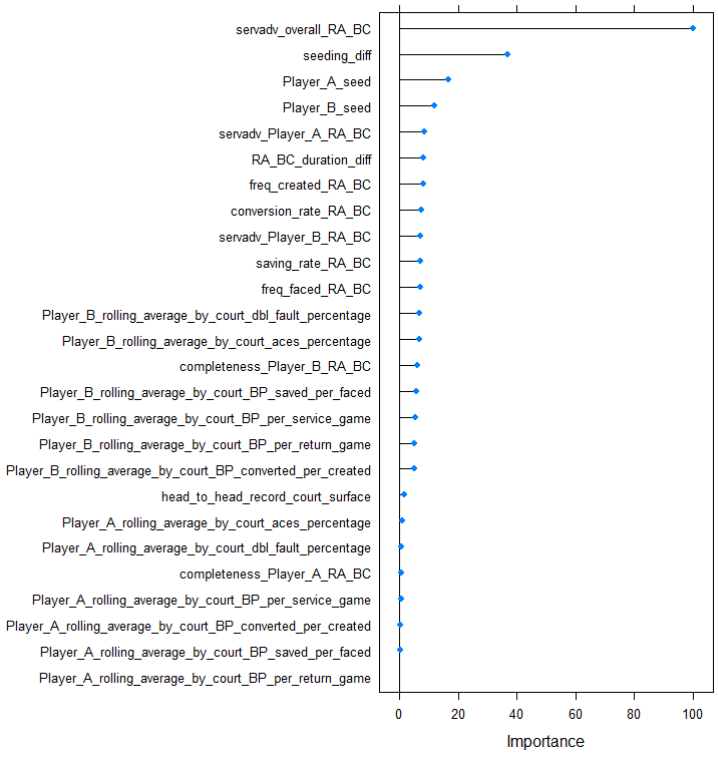


Figure C.6: Engineered Features Historical Average By Court

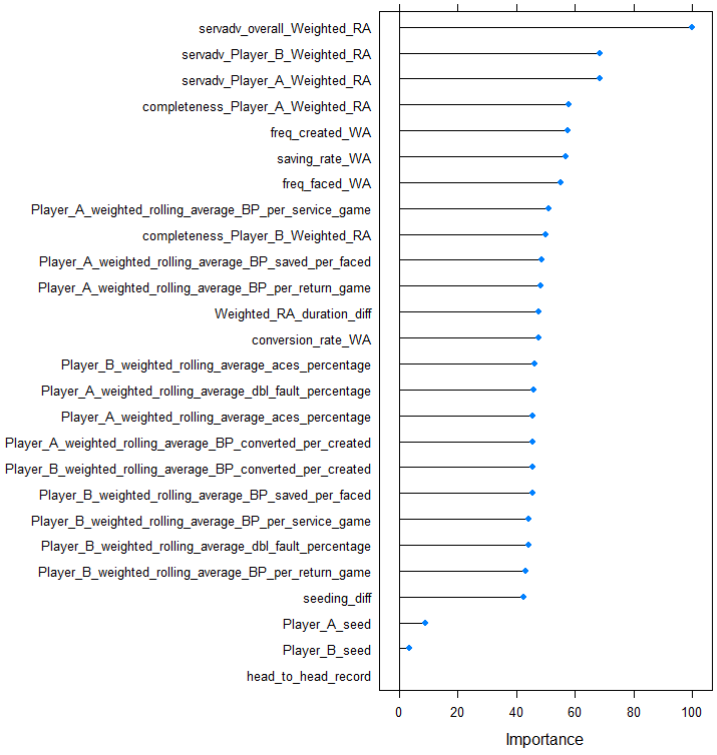


Figure C.7: Engineered Features Time Discounted Historical Average

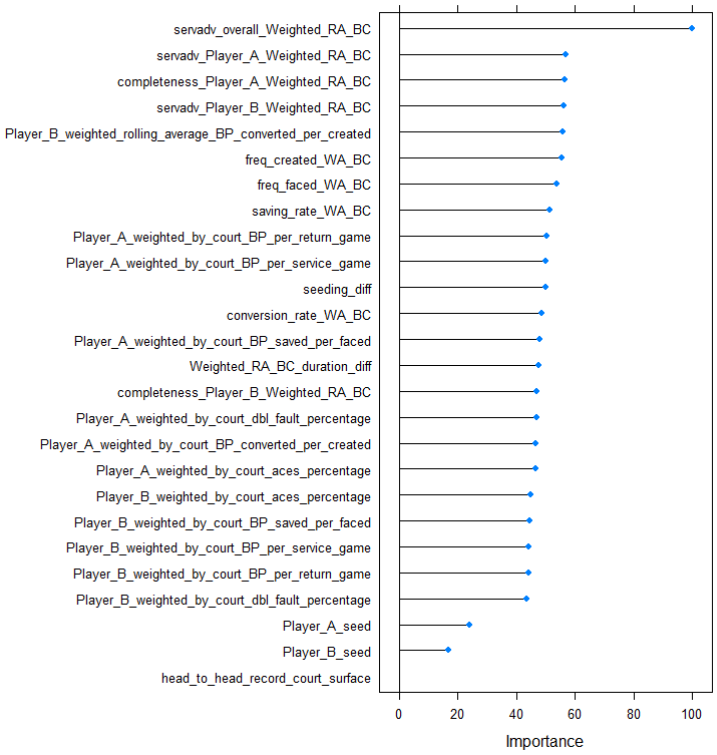


Figure C.8: Engineered Features Time Discounted Historical Average By Court

Appendix D

Gradient Boosting

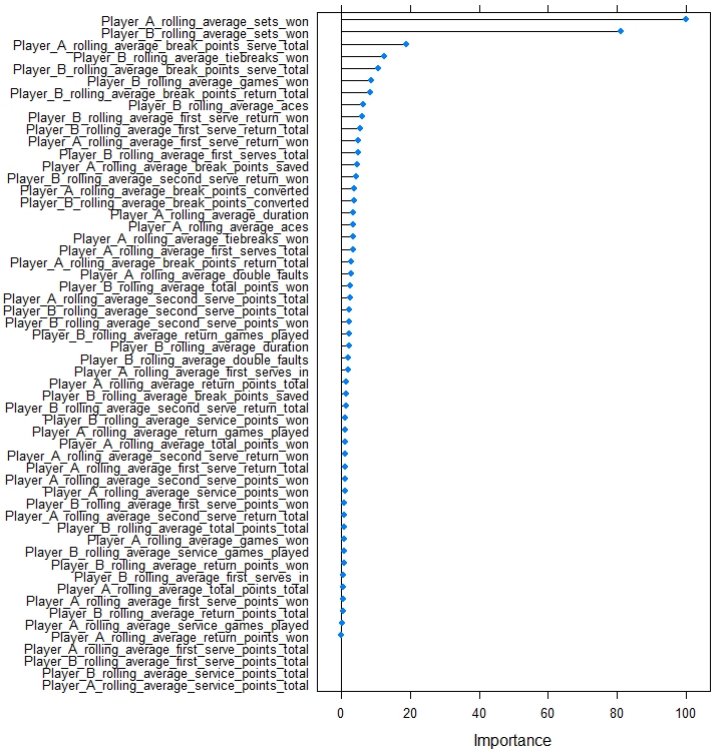


Figure D.1: Raw Features Historical Average

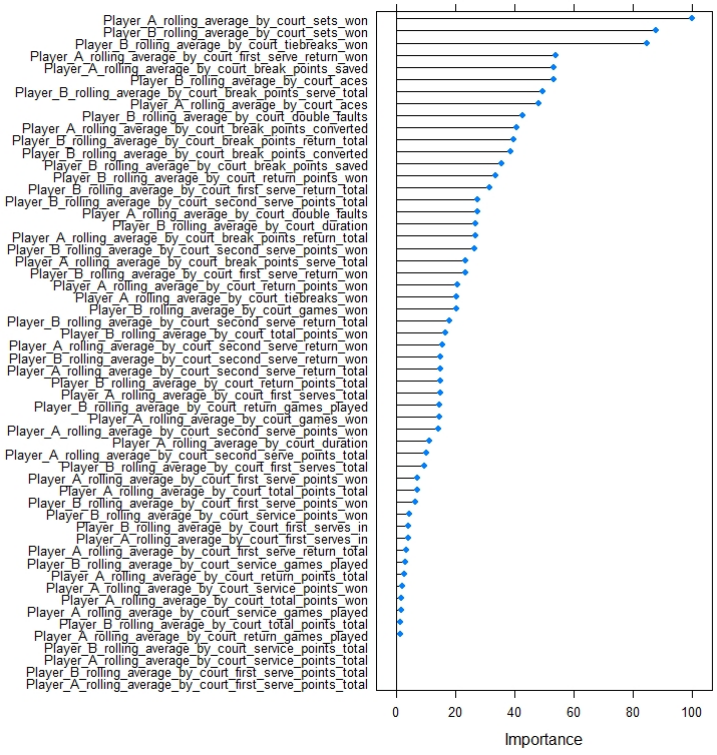


Figure D.2: Raw Features Historical Average By Court

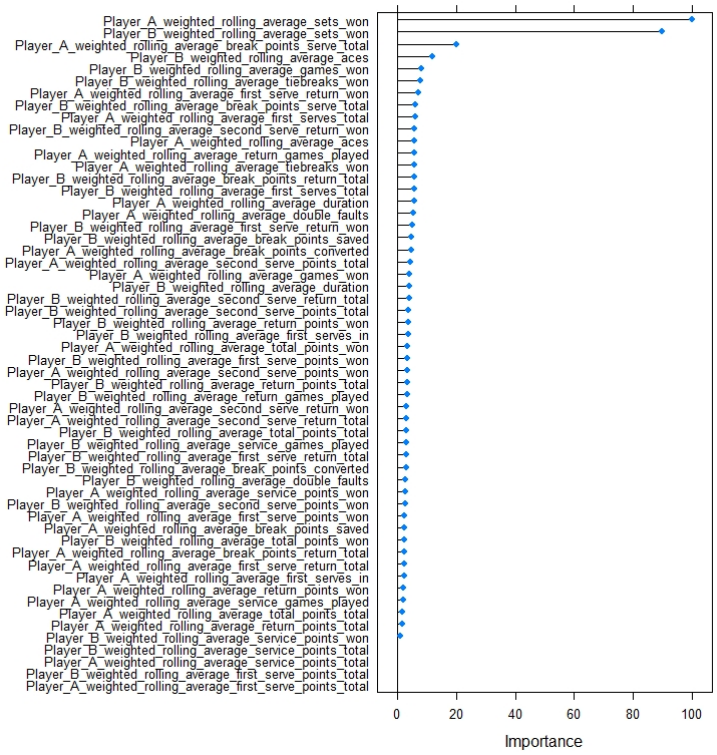


Figure D.3: Raw Features Time Discounted Historical Average

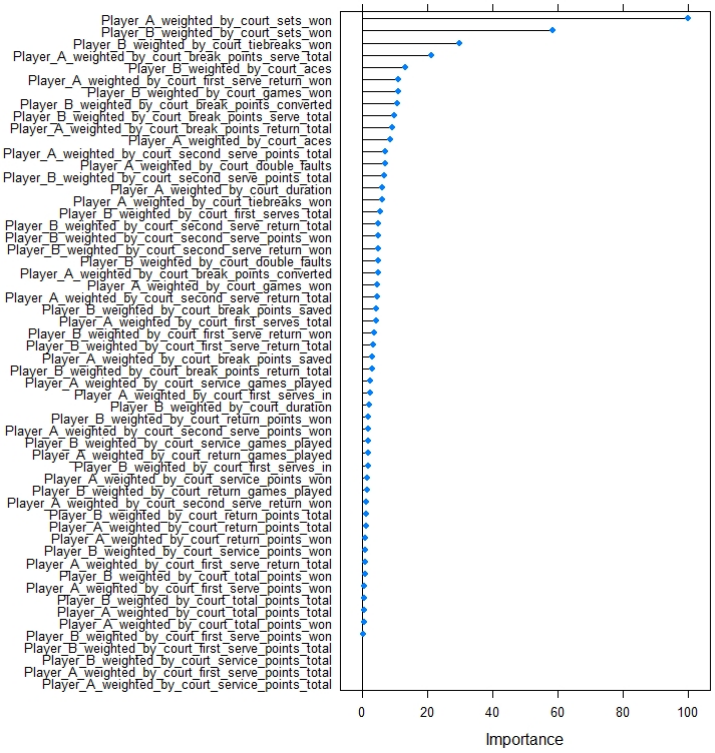


Figure D.4: Raw Features Time Discounted Historical Average By Court

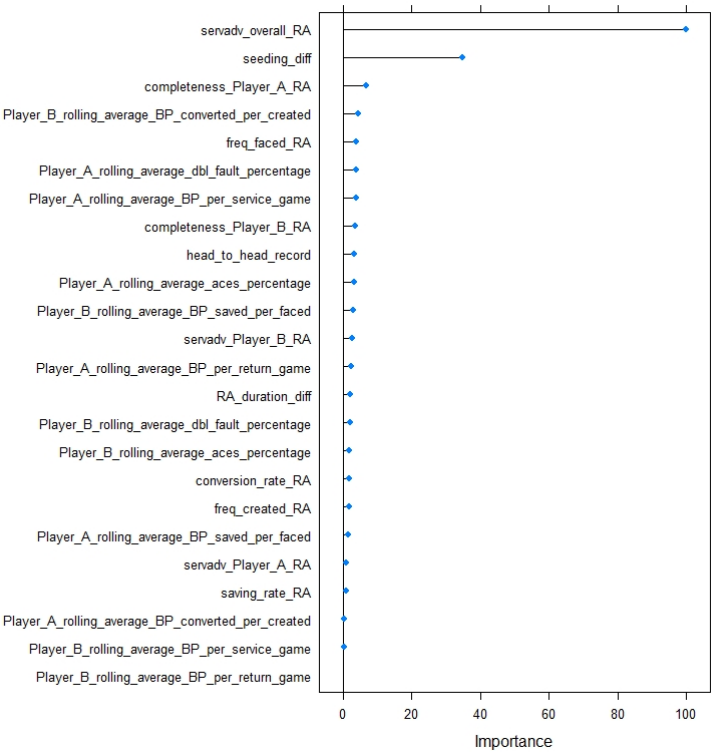


Figure D.5: Engineered Features Historical Average

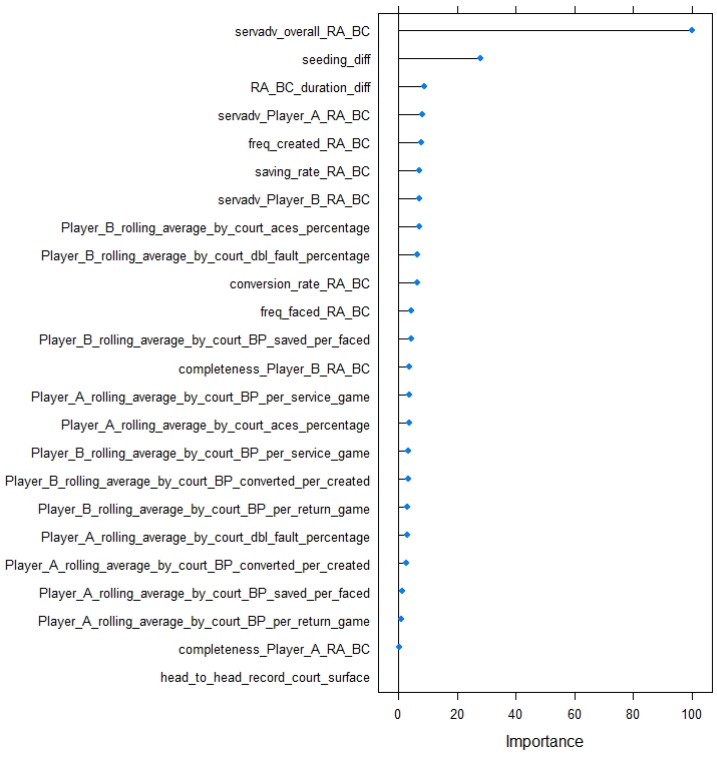


Figure D.6: Engineered Features Historical Average By Court

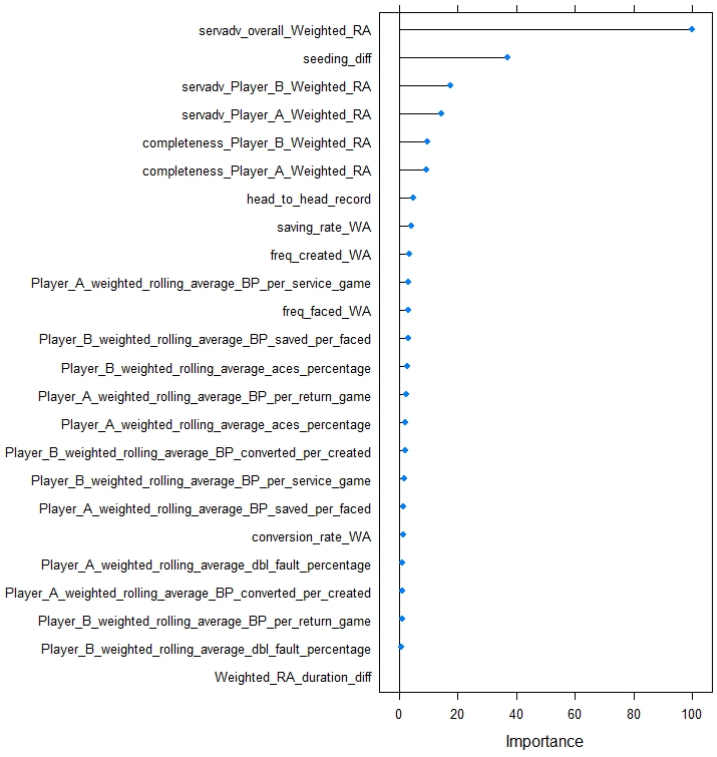


Figure D.7: Engineered Features Time Discounted Historical Average

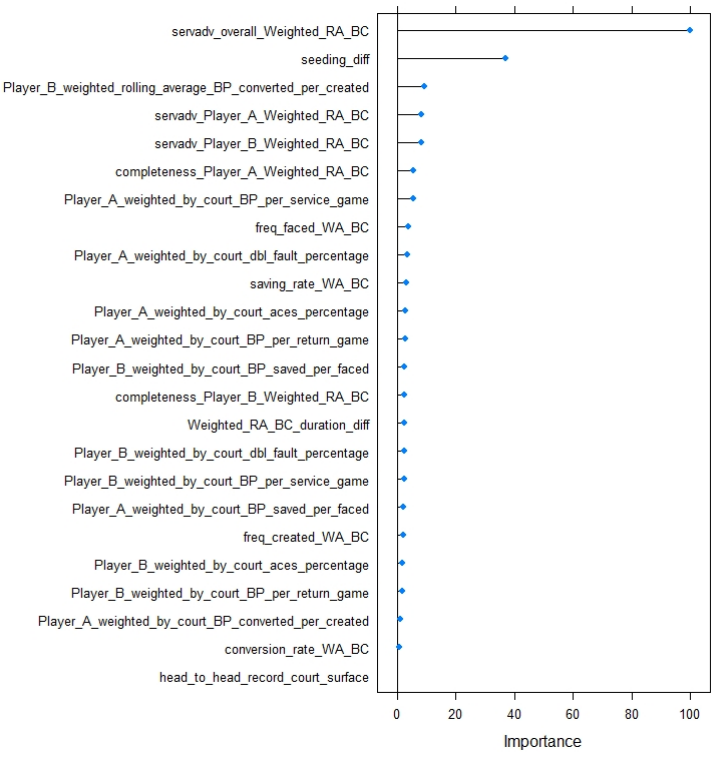


Figure D.8: Engineered Features Time Discounted Historical Average By Court

Appendix E

R Code

The full R code used in this project can be found on GitHub.