Exploration of Throw-Ins in Soccer Using Machine Learning Algorithms

Andreas Bancheri

A Thesis

in

The Department

of

Mathematics and Statistics

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Science (Mathematics) at

Concordia University

Montreal, Quebec, Canada

November 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Andreas Bancheri

Entitled: Exploration of Throw-Ins in Soccer Using Machine Learning Algorithms

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Mathematics)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

______________________________ Thesis Co-Supervisor

Dr. F. Godin

______________________________ Thesis Co-Supervisor

Dr. J. W. Smith

______________________________ Examiner

Dr. S. Brugiapaglia

______________________________ Examiner

Dr. A. Sen

Approved by ______________________________

Dr. Y. Chaubey (Graduate Program Director)

____

Dr. P. Sicotte (Dean of Faculty)

____

Date

Abstract

The evolving field of sports analytics is still in the early stages of its adoption. Moreover, soccer analytics utilizing tracking data is even further limited. This research is motivated by Liverpool's integration of a department for throw-in research, assisting them in winning a league title. This research project makes use of the (generously provided) German national soccer team (DFB) tracking and event data which includes all player movement during a game, and more specifically, movement before and after a throw-in. The probability of a throw-in being completed (according to two mutually exclusive definitions) is estimated using various metrics developed using the aforementioned tracking and event data. Binary classification models are used to estimate the completion probability of a given throw-in. The model can be applied to many practical questions centered around throw-in success along with providing use to soccer match analysts.

Acknowledgements

I would like to express my deepest appreciation to my two supervisors, Dr. Frédéric Godin and Dr. Joshua Wyatt Smith. First, thank you to Dr. Godin for being so caring and helpful to all students. Especially however a very resourceful supervisor for all aspects of the mathematical components along with providing immense advice to formal writing and proper scientific presentations. Second, thank you to Dr. Smith for introducing me to the world of sports analytics, and more precisely soccer analytics, which is one I have dreamt of working in since childhood. Without your knowledge on all things data science, sports, and the interaction of the two, none of this would be possible. Thank you both for the guidance and truly challenging and driving me into the final product which it has become. I am grateful for everything and excited to see what this can become in the future.

Thank you to the DFB, and more specifically Dr. Pascal Bauer, for allowing this research project to become a reality. Without the guidance, meetings throughout the duration of the project and the data provided, none of this would be possible.

Finally, thank you to my parents and two brothers for always being so supportive. Many long nights went into this and without their motivation on some of those difficult nights, none of this would be possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Big data and analytics have become an integral part of society with applications in the fields of finance, engineering and health. The sports industry is a 440 Billion dollar industry [1] so it was only natural that those participating in this field would seek any possible opportunity to improve their team's success (which in turn would increase their revenue). The development of the box score by Henry Chadwick, the father of baseball, dates back to 1859 [2] while there are records of scorecards in cricket going back to 1746 [3]. The economics of baseball were first analyzed by Simon Rottenberg [4] and George Lindsay [5] in their respective Operations Research papers and Stefan Szymanski in his article in the Kinesiology Review stated that "baseball is a game well suited for operations analysis, and is already well provided with statistical records of past performances of individuals" [6]. Books published by Bill James in the 1980s are some of the first examples of using player statistics to look at their potential predictive power [6]. These books were largely credited with the creation of the field of sabermetrics, defined in 1980 by James as "the search for objective knowledge about baseball" [7].

In 2003, as Michael Lewis published *Moneyball: The Art of Winning an Unfair Game*, the role of data in decision making in sports was forever changed. Put broadly, the book was about identifying the inefficiencies that baseball was creating and trusting the objectivity that the data provided. The book details the story of the 2002 Oakland Athletics, which, shocked the baseball world. After winning 91 games in 2000 and 102 games in 2001, they had just lost their top three players to free agency and were expected to be the bottom of the league in 2002. Instead, Oakland Athletics won 20 consecutive games, setting an American League record and winning the AL West with a record of 103 wins and 59 losses. They later lost in the first round of the playoffs, but the season was a surprise nonetheless.

Although sports analytics was an area of research already at this time, this phenomenon propelled teams of all different sports to further fund and utilize a data analytics department. Soccer of course was involved with this movement and over the past two decades it has made immense improvements.

However for all the research and improvements that was being made, one key aspect seemed to be forgotten, set pieces[1], and more specifically, throw-ins. T his dramatically changed however after Liverpool's shocking 2019-2020 season. Thomas Grønnemark, was hired during the off season of the previous season as a throw-in coach and was integral to their 2019-2020 Premier League title. "*A season later , Liverpool improved their throw-in retention under pressure percentage from 45.4% to 68.4% and had risen from 18th in these standings to first in the Premier League.*"[2] Furthermore "*Liverpool scored 14 of their 85 Premier League goals from throw-in situations in their title-winning 2019-20 campaign*", which is an astounding 16% of all their goals that season. The topic of throw-ins has not received extensive attention in the literature, and additional research on the topic is well needed.

An exception is the recent paper from Stone, Smith, and Barry [8], which analyzes 16,154

[1]Set piece in soccer is any time the ball is returned to open play; penalties, corners, free-kicks and throw-ins.

[2]https://www.espn.com/soccer/liverpool-engliverpool/story/4172876/meet-liverpools-secret-weapon-throw-in-coach-thomas-gronnemark

throw-ins from the 2018-2019 Premier League Season; a remarkable 8.8% of throw-ins led to a shot at goal. In comparison to the 16% rate scored by Liverpool, this is a stark difference. In one year of training Liverpool was able to see large improvements and fast results. This is an indication that more professional teams should be adopting the practice of analyzing throw-ins.

In this thesis, soccer research and more specifically throw-in success is analysed with the goal of creating a model which can produce an expected completion percentage for a given throw-in. This topic is one with little academic research and this paper aims to fill it.

Throw-in success is modeled using binary classification models in order to estimate completion percentage to any potential receiver for a given throw-in. The research begins by discussing the data and all of its intricacies in Chapter 2. In this chapter all of the model features were described, from raw data and the development process to create them into data that can be fed into the models described in Chapter 3. Along with the classification models, the model performance metrics that are used in this research paper are explained in this chapter. Chapter 4 discusses each of the respective model results along with further discussion. Finally, Chapter 5 discuses all of the practical applications of this work and how it can help some current inefficiencies in soccer throw-in strategies. The results show that the model provides an encouraging framework of achieving the goal of a universal throw-in metric. Therefore, any given throw-in may be evaluated, providing a meaningful tool to soccer teams, in the footsteps of xG (expected goal) or xPass (expected pass) models.

Chapter 2

Data

2.1 Overview of Soccer Data

Technology from within the sports world is constantly improving and one of the biggest innovations is the integration of technology called Computer Vision. Computer vision is the simulation of human vision that is meant to train the computer to understand and interpret vision of the world using several algorithms and applications that support this science [9]. The main use of computer vision, as applied to soccer data, is player detection and tracking throughout the match. The use of their movement is invaluable when developing soccer research. Unlike many other sports, many raw statistics occur due to the infrequent nature of scoring in the match (as opposed to American football, basketball or hockey for example). Without these raw statistics, soccer research has turned to in-depth topics using this tracking data such as lineup chemistry or counter-attack effectiveness. Player detection and tracking are fundamental elements required for creating such tasks which provide a better understanding of the match. Player detection and tracking are quite difficult asks due to many difficulties in accumulating the actual data given the many occurrences during the match such as, similar appearance of players, complex interactions, unconstrained outdoor environment, changing background, varying number of players with unpredictable movements, abrupt camera motion and zoom, calibration inaccuracy due to the low textured field and edited broadcast video, noise, lack of pixel resolution especially on small distant players, clutter and motion blur [10].

As is the case with most professional soccer competitions (at the club and country level), teams collect positional and event data in a pre-defined and consistent format [11]. Player tracking systems in soccer were first used in the late 90s and have been constantly improving with developments in camera and image processing [12]. Positional data, also known as tracking or movement data [13], provides the positions of all players, referees, and the ball in a (X,Y) coordinate system on the field with a frequency of 25 Hz. The positions of all 22 players and the ball are accurately gathered by an optical tracking system, which captures high-resolution video footage from different camera perspectives [11]. A data vendor, named TRACAB [14], achieves real-time and high precision tracking thanks to advanced camera configuration, hardware developed and stereo vision technology[10]. The tracking data which is received contains the (X,Y) coordinate of all 22 players and the ball every 0.04 seconds. Event data is manually obtained by operators who are at each match and are recording such data. Event data for soccer includes every action that occurs in the match such as a pass, dribble, foul or throw-in.

2.2 Data Used

The data for this research project was generously provided by the German Football Association, also known as the DFB. Tracking and event data for each international match that the German National Soccer team participates in, from U16 to the Senior team for both men and women are recorded. The event data was collected according to the official DFB match-data catalog[1], and the optical tracking data was provided by the above mentioned TRACAB system[2].

2.2.1 Data Processing

The tracking data that the DFB provides (from the TRACAB [14] vendor) is invaluable due to the difficulty of developing and acquiring it. One of the initial challenges at the start of this research project was to identify how the data can be utilized to its fullest potential and to extract all pertinent and useful information from it to run future models. Although throw-ins may occur at similar spots of the field, the interactions of the two teams can be different between these throw-ins and therefore the power of the tracking data can be examined and used in analyses. This can be explicitly seen in Figure 2.1. In both throw-ins, the blue team throws to number 20, however the defensive coverages (and offensive positions) are vastly different in the two throw-ins. This information is not covered in event data and consequently not taken into consideration when building models and analyses purely from such data. Over 6000 throw-ins, coming from all levels of international competition were evaluated for this project given by the DFB.

2.2.2 Raw Data

Before discussing how the features are developed using the raw data given by the DFB, the actual format of the data provided will be discussed in order to provide some clarity for the subsequent steps and sections. The exact format for all the event data provided is shown in the Appendix in Tables A.1 and A.2 along with explanations for each of the variables. The tracking data which was explained above in Section 2.1 is a tuple of 8 attributes for the

[1]https://s.bundesliga.com/assets/doc/10000/2189_original.pdf
[2]https://chyronhego.com/wp-content/uploads/2019/01/TRACAB-PI-sheet.pdf

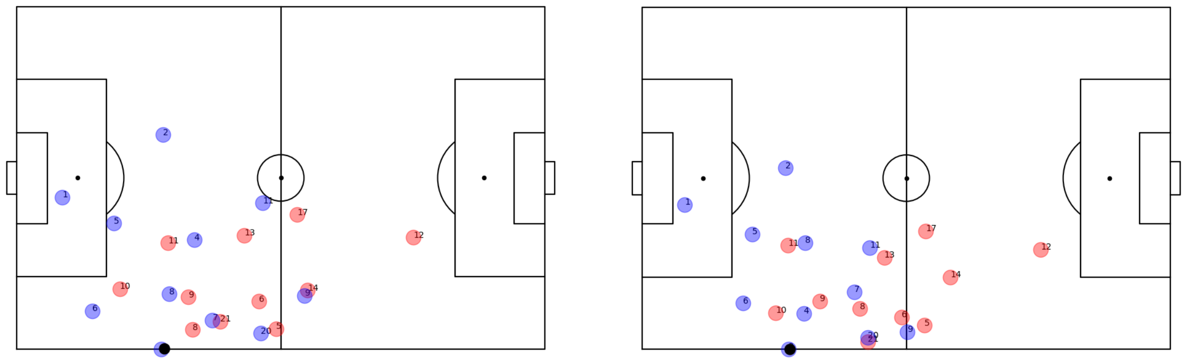

Figure 2.1 – Player positions of two throw-ins from roughly the same location - but very different environments. The player positions and the interaction between the throwing team and defending team are vastly different. In both cases, the blue team is playing from left to right.

player data and a tuple of 9 attributes for the ball data. The tracking data for the player tuple consists of the tracking data index, TeamID, PlayerID, distance covered (from frame to frame), minute of the match, half of the match (first 45 minutes vs second 45 minutes of match time) and the X, Y coordinates. The ball data tuple consists of the tracking data index, ball possession (which team is in possession of the ball for that frame), ball status (in or out of bounds), distance covered (from frame to frame), minute of the match, half of the match (first 45 minutes vs second 45 minutes of match time), X, Y and Z coordinate[3].

2.2.3 Feature Creation

Features[4] are created through a transformation of the raw data explained above, using both tracking and event data. The idea for what features to use was a combination of providing those used in the Expected Goals (xG, [11]) model along with inituitevely thinking of others (such as Pressure and Time to Take a Throw-in). Before being able to apply the predictive analysis algorithms developed, raw data had to be transformed into a matrix format con-

[3]Defined as the third-dimensional coordinate, i.e. the distance above the playing field.

[4]A model feature is a quantity taking the entire information at a time point and summarizing it into a single quantitative variable.

taining the calculated features along with the considered response variables[5]. Each column of the tabular matrix corresponds to a feature along with the response variables, with each row corresponding to one instance (one throw-in). A description of all features can be found in Table 2.1. To make full use of the synchronization of the data sources, the features are based on both event and tracking data. The features are independent in creation, although dependencies can be found as shown in Appendix A.4, Figure A.22. Scatter plots for each combination of features is shown, and it is clear that dependancies do exist between the features.

Feature	**Value**	**Description**
Distance to the potential receiver	Numerical	The Euclidean distance from the position of the thrower to the position of the potential receiver (in meters).
Angle to the potential receiver	Numerical	The angle defined on a 0-180° scale, normalized from the thrower to the potential receiver. Figure 2.2 illustrates.
Match Time	Numerical	Time of the match when the throw occurs.
Score difference	Numerical	Score difference from the offensive (throwing) team's perspective at the time of the throw.
Time to take the throw-in	Numerical	Time taken from the ball going out of bounds to the throw-in.
Receiver movement	Numerical	Total distance travelled by the potential receiver from time of the throw to time of reception.
Location on the field	Numerical	X-Coordinate of the throw.
Pressure on the potential receiver	Numerical	Pressure on the potential receiver according to [15]. Described in detail below.

Table 2.1 – Features derived from tracking and event data used to train the models developed.

The calculations for receiver movement, and more specifically pressure, are described in further detail below. Receiver movement and pressure utilize tracking data. The tracking data provides the Euclidean distance each player (and the ball) travels from one frame[6] to

[5]The response variables will be explained in more detail in the following section.

[6]A frame is defined as the moment in time that the positional data is recorded - occurring every 0.04

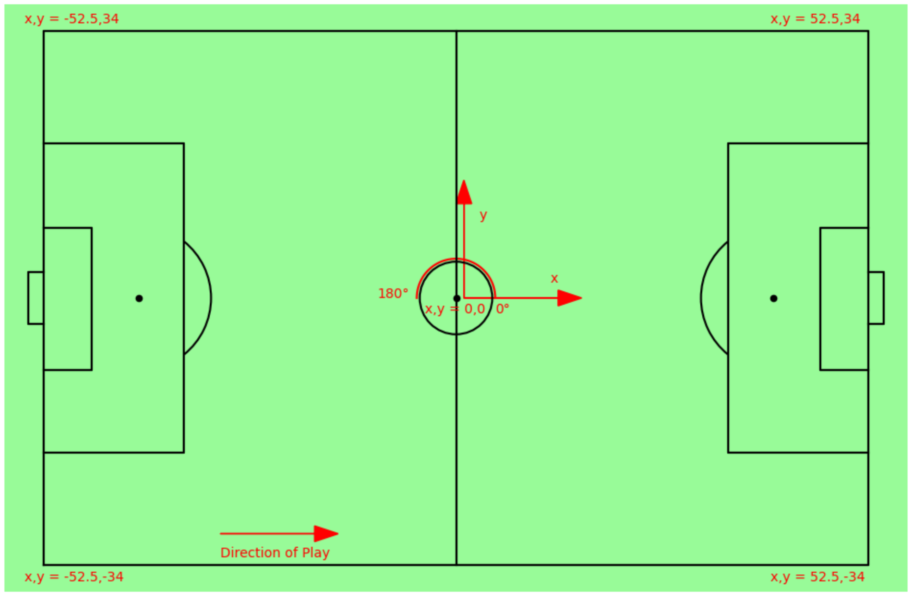

Figure 2.2 – Dimensions and orientation of the field for all the data used in this project. All data is automatically normalized for the attacking team to be playing from left to right (as shown with the direction of play arrow).

the next. This distance was calculated by taking the difference between the location at the time of the throw-in to the time of reception for the predicted or known receiver.

The pressure metric used in this research project is derived from the logic written in the paper by Andrienko et al. [15]. The methodology which he developed can be used to estimate the pressure exerted at any point on the field. As applied, in this research project the pressure exerted on the potential receivers from a throw-in is of interest. The ideology of pressure is important as it leads to the strategic concept in soccer analysis which has come to be known as *pressing*. Pressing is defined as a tactic created by the defending team in which they intend to pressure the opponent which possesses the ball. The goal of using this tactic is to win the ball, or at least, deprive the opponent of the opportunities to develop an attack. Effective pressing has been key to the success of several high-level football clubs, such as FC Barcelona, Borussia Dortmund, and Atletico Madrid. [15]. Given these factors, when considering this technique from the offensive (throw-in team's perspective) one would be

seconds.

inclined to want little pressure on our potential receiver as possible to increase the expected completion rate.

Following the framework of Andrienko et al. [15], define a *pressure target* as an area on the field of interest in which the pressure being produced will be measured. This can be anywhere on the field with most common studies looking at the ball or players as the pressure target. Consider the direction from the pressure target towards the goal that is attacked or towards an attacking team player as the *threat direction*[7]. Let V^{threat} be the vector originating from the pressure target towards the threat direction (threat defined as the opposing team's net direction), which can be simplified as the x-axis. Consider V^{TP} to be the vector from the pressure target towards any potential pressers. In Figure 2.3, this is the vector from the receiver (the red X) to all the defenders (blue dots). In addition, the pressure target (the receiver) is easily visualized along with the threat direction, which is the direction towards the opposing net (left to right).

The calculation relies on two main formulas. The first equation creates a *distance limit* (L), which approximates an oval shape in polar coordinates (θ,L) - creating a *pressure zone*. The boundary of the pressure zone is a parametric curve with two parameters: front distance and back distance. D_{front} is the limit for exerting pressure when the presser is in front of the pressure target (i.e. when $\theta(V^{threat}, V^{TP}) = 0$) and D_{back} is the limit in the case when the presser is behind the pressure target (i.e. when $\theta(V^{threat}, V^{TP}) = \pm 180$). Andrienko et al. [15] chose values of $D_{back} = 3$m and $D_{front} = 9$m after consulting with soccer experts and running various tests with computer scientists. The soccer experts are former professional soccer players (not named) from various European soccer leagues. The distance limit for the pressure algorithm is given by:

$$L = D_{back} + \frac{(D_{front} - D_{back})(z^3 + 0.3z)}{1.3} \tag{2.1}$$

[7]The threat direction is always to the right, due to normalization explained in Figure 2.2.

where $z = \frac{1+cos\theta}{2}$.

The maximal theoretically possible pressure (100%) is when the presser is exactly in the location of the pressure target. For any point within the above mentioned *pressure zone*, the pressure is estimated as:

$$Pr = (1 - d/L)^q * 100\% \qquad (2.2)$$

where d is the distance of the point to the pressure target and L is the distance limit determined by Formula 2.1. The exponent q regulates the speed of the distance decay, i.e., how fast the pressure decreases with distance. Andrienko [15] deems $q = 1.75$ optimal after various tests, as explained in his paper.

As a result, the final *pressure* value which is assigned to each of the potential receivers (and subsequently used to power all of the models), is the sum of all of pressers pressure value (Equation 2.2) in the receivers given zone. Figure 2.3 illustrates how the *pressure* metric is obtained for a single receiver - this algorithm is then computed for each receiver (all the red circles in the figure).

As the pressure metric is cumulative[8], as given by Equation 2.2, there is no theoretical maximum value of the metric (expected to have values greater than 1).

2.2.4 Defining a Success

Although the TRACAB [14] data provides a column of data for the binary outcome of each throw-in, many issues arose when taking a closer look at what constituted a truly successful throw-in. Event data (mentioned above in Section 2.2) has an *Evaluation* column which is manually entered by a match operator after each event in the match. A match operator is an employee of the data provider who is physically located at the match and manually

[8]The summation of all of pressure applied by pressers in the receivers area.

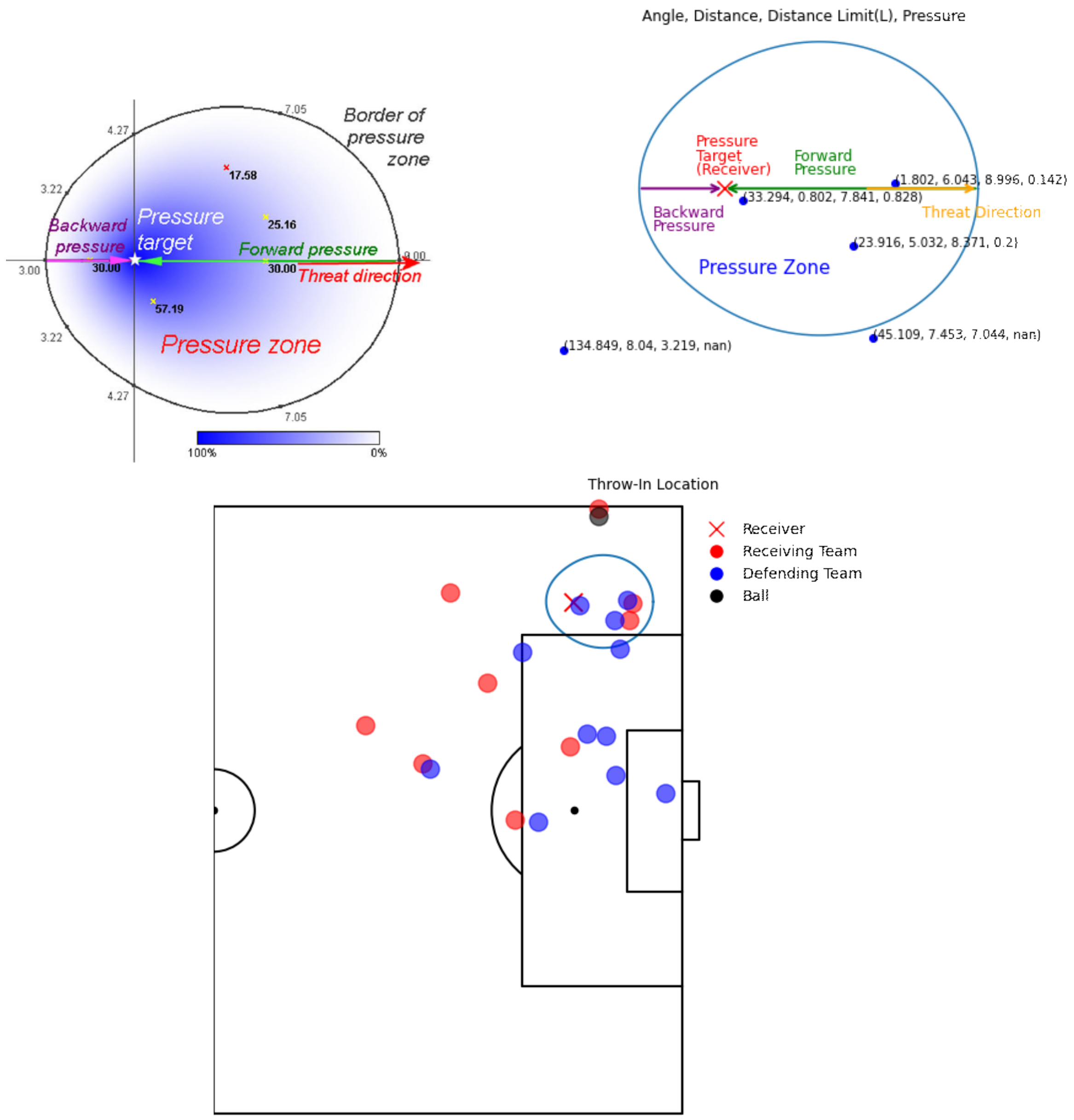

Figure 2.3 – Illustrations of the image pressure framework described, following [15]. The top figure is taken directly from [15] showing a pressure bubble. The upper right image is an example of a throw-in from the data set analysed for this research, using pressure values from the algorithm created. The bottom image is the throw-in from which the pressure values are calculated, showing the amount of pressure each defender is providing for this throw-in.

marking each event as a success or failure and verifying time occurrences of them. The *Evaluation* column identifies as three different classifications: *Successfully Complete*, *Success* and *Unsuccessful*. The *Success* classification is assigned when a foul occurs, but the throw-in team maintains possession and/or thrown to the unintended receiver. Problems were found when watching throw-ins from the data (using match footage, which the DFB provided) and realizing some of the throw-ins deemed successful should not have been. The main issue emerged when realizing that for throw-ins in which a lot of chaos[9] occurs and no clear possession is made, the match operator designates the success based on *first touch*.

This is an issue as first touch does not necessarily mean sustained possession of any kind, which is ultimately the objective from your throw-in (maintaining possession means more chances to score and less chances to concede). This is not universal and not always the case however, and this will be explored in more depth below. This issue was brought up to the DFB and it was suggested to modify the way throw-in success was being classified due to the reasoning mentioned above.

The suggestion was based on the research from Stone [8] with two clearly defined types of success: *first touch* and *threshold retention*. Successful *first touch* is defined as: A player from the same team which throws the ball into play makes first contact with the ball post throw-in without an opposition player making contact. Successful *threshold retention* is defined as: The ball is retained in possession[10] for 7 seconds from the point in which the ball is thrown.

Taking Stone et al. [8] research into consideration, two different success labels were created for the analyses. They are considered *first touch* and *threshold retention* success following the same definitions outlined above. The results will be further discussed as the two defini-

[9]Such as the ball bouncing off players or a lot of fighting for the ball

[10]The time (seconds) from the throw-in action to the end of possession. A possession is defined as a passage of play during which one team is largely in control of the ball. This may involve that team temporarily being dispossessed, but a new possession will only start if the opposing team is then able to demonstrate that they are fully in control of the ball.

tions are compelling as they create practical applications for the coaches/players in different scenarios. For example, in the offensive half (close to the opposing net), maintaining possession seven seconds into the future is not a huge concern as one is most likely shooting on net within that time. On the contrary, throwing the ball to the defensive half and maintaining possession is critical as it is the mirror image of the scenario previously mentioned. If one loses the throw-in, chances are the opposing team will be making a dangerous play on your net in the following events. The success rates for the 6119 throw-ins using this method were, 83.8% for *first touch* success (5128 instances) versus 65.6% for *threshold retention* success (4012 instances).

2.2.5 Threshold Time Selection

This section will explore the approach used to quantify a successful possesion along with the subsequent reasoning as to why the 7 seconds was used as an objective stopping point. Stone et al. [8] also used 7 seconds, thus it was an interesting threshold to use in that comparison can be drawn with the research of that paper.

The methodology began by using the tracking data for each throw-in and checking how many frames the throwing team maintained possession. This can be done using the tracking data (explained in 2.2.2). Within the data is a column at each frame which indicates the team in possesion. Looking at this possession column at each frame after the throw, the length of time one maintains (or loses) possession can be checked by looking at when the possession column remains the same (or changes to the other team).

As previously mentioned in Section 2.2.3, the features were built on two games worth of tracking data and this method of selecting the threshold time was created using those same two games. A frame (as explained in Section 2.2.3) is defined as the moment in time that the positional data is recorded - occurring every 0.04 seconds. The figure below plots the

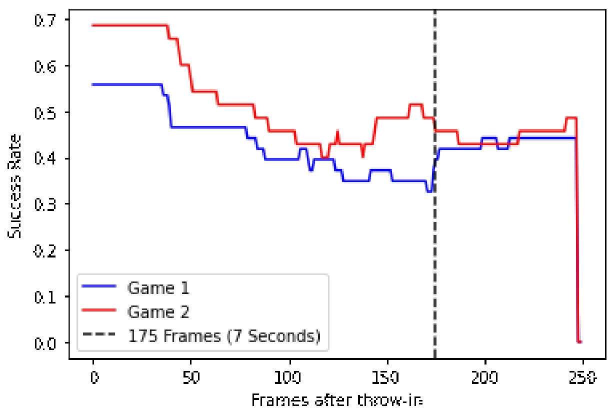

Figure 2.4 – Frame by frame success rates post throw-in. Game 1 contained 43 throw-ins and Game 2 contained 35 throw-ins.

percentage of the time that the throwing team maintains possession from the instance of the throw-in to 250 frames after the throw-in is thrown (10 seconds).

At each of the 250 frames, the 78 total throw-ins are evaluated to see how often the throwing team maintains possession and the success rate plotted on Figure 2.4 is the average over all throw-ins of that individual frame position. The results stemming from this ideology in Figure 2.4 was encouraging as the pattern for the two games follow a similar trend. A clear decreasing trend occurs until about 150 frames at such point the success rates seem to stabilize and past 200 frames it is close to 50% until the end of the plot.

Intuitively, along with watching game film, an explanation as what is occuring can be explained. After a certain amount of time once the chaos of a throw-in occurs, either one team or the other possesses the ball and thus the success rate of the plot stabilizes. The selection an exact frame (time) to end the threshold will never be perfect but given the plot results along with the rationale which explains it, 7 seconds provides a concrete threshold selection.

2.2.6 Data Cleansing

All the features mentioned in the previous section were individually engineered using the event and tracking data provided by the DFB. Several issues arose throughout this process. One of the more difficult issues being identifying data errors from the provider and providing solutions to rectify them.

The first issue in this process was to correctly identify when (what frame) a throw-in was received and the subsequent timing by the receiver of the ball. As mentioned in the above section, the receiver is known when a throw is deemed *SuccessfullyComplete*, however for *Successful* and *Unsuccessful* throw-ins an issue arises as to not knowing who (or when or where) the receiver is at the time of reception.

For the *Unsuccessful* throw-ins, the method applied in this project is to verify where the ball is two (2) seconds in the future and to use the closest player nearest to the ball as the receiver. Using this method, the frame along with the player location were found. This method was only necessary for the throw-ins with an incorrect receiver or none at all. For the throw-ins with the correctly tagged receivers, the closest receiver to the defender at the time of reception was found and used as our intended receiver (and subsequent feature values derived from it).

Similar to the first issue, a second issue was deemed to require correcting, which is that the distance value (for the throw-in location to receiver) from the data provider was also often incorrect. In many instances, the data provider was incorrectly determining the receiver's distance[11] and, as a result the distance metric was developed to correct this issue. The *distance* metric calculated using the developed algorithm discussed above is used to determine who the intended receiver was (as opposed to using the provider's).

[11]Mainly due to the issue mentioned in the paragraph above, where the *Successful* and *Unsuccessful* throw-ins were tagging the incorrect receivers.

One match was also removed from the data set as it was deemed completely faulty due to an issue with the data provider. Various other exclusions were also applied to the data due to these mislabeling issues. When the *distance* feature value was calculated to be over 40 meters, they were individually inspected using the match footage to obtain a better understanding of the sequence of events for the given throw-in. Most of these scenarios were due to the event not being a throw-in but rather a goal kick[12], thus being mislabeled by the data provider and correctly being removed from the data set.

[12]A goal kick in soccer is when the goalkeeper is given the ball to kick out of the goal area due to the other team having kicked it over the touch line on either side of the net.

Chapter 3

Model Development

3.1 Machine Learning Classification Theory

The objective of this chapter is to use supervised learning algorithms to create a predictive throw-in completion model, utilizing both event and tracking data. Supervised learning algorithms are tasks which map each observation of the predictor measurement(s) x_i, $i = 1 \ldots n$ to an associated response measurement y_i [16]. In statistical literature, the inputs are often named the predictors or also more commonly known as the independent variables. The outputs are named the responses or also more commonly known as, the *dependent variables* [17]. The objective is to fit a model that relates the data responses to the predictors, with the aim of accurately predicting responses for future observations.

Variables can be characterized as either *quantitative* or *qualitative*. Quantitative variables take on numerical values, while qualitative variables take on values in one of many different classes or categories. The distinction in type of variable being examined has led to two different types of problems for supervised learning: *regression*, when predicting quantitative outputs, and *classification*, when predicting qualitative outputs [17]. Care must be taken when making the not so simple distinction between Least Squares Linear Regression which is used with a quantitative response and, Logistic Regression which is typically used with a qualitative (two-class, or binary) response [16]. In certain situations, statistical methods such as K-nearest neighbors and boosting, can be used used for either quantitative or qualitative variables. Binary classification is the specific case when there are only two classes or categories such as the commonly used variables "success" or "failure" (numerically inputting the values 0 for failure and 1 for success). In summary, statistical models assume the relation between the response y_i and explanatory variables $\{X_{i,1}, \ldots, X_{1,p}\}$ is characterized by some function f.

Quantitative relationship example:

$$Y_i = f(X_{i,1}, \ldots, X_{i,p}) + \epsilon_i \tag{3.1}$$

where we have p explanatory variables and some ϵ_i noise term.

Qualitative relationship example:

$$\mathbb{P}[\mathbb{G}_i = g | X_{i,1}, \ldots, X_{1,p}] = f(X_{i,1}, \ldots, X_{1,p}) \tag{3.2}$$

where $\mathbb{G}_i$ represents a categorical output (class label, defined as g in the above example).

3.1.1 Linear Models and Least Squares

Linear models are one of the most important aspects of statistics due to their many and powerful uses. Consider a regression setting, given a vector of quantitative inputs $X = \{X_1, \ldots, X_p\}$, which we predict the output of Y using the following model:

$$\hat{Y} = \hat{\beta}_0 + \sum_{i=1}^{p} X_i \hat{\beta}_i = X^T \hat{\beta} \tag{3.3}$$

where $\{\beta_0, \dots, \beta_p\}$ are the coefficients of the linear model. The model in vector form is an inner product of the transposed input matrix (X^T) times each of the corresponding coefficients (β). If we consider $\hat{Y}$ as a function over the p-dimensional input space, $f(X) = X^T\beta$ is linear. Our coefficient parameters ($\beta_0, \dots, \beta_p$) can be chosen by a method known as the method of *least squares*. In this approach, coefficients are selected which minimize the residual sum of squares[1]:

$$(\beta_0, \dots, \beta_p) \in \underset{(\beta_0,\dots,\beta_p) \in \mathbb{R}^P}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.4}$$

where $\hat{y}_i$ follows Equation 3.3 above. The solution to the OLS Equation 3.4 is given by $\hat{\beta} = (X^T X)^{-1} X^T y$, where the fitted value at the ith input x_i is $\hat{y}_i = x_i^T \hat{\beta}_i$. Linear regression provides a smooth, (and as the name suggests, linear) fit, yet issues can arise. The linear regression model is sometimes too inflexible to accurately represent the pattern between the response and predictor variables.

To gain flexibility, a common approach is to create a transformation of the predictors set. This can be completed by using a parametric approximation to our function f (of our input vector X) referred to as *Linear Basis Expansion*:

$$f_\theta(x) = \sum_{k=1}^{K} h_k(x)\theta_k. \tag{3.5}$$

The functions h_k transform the input vector X. There are many possibilities for this function such as polynomial expansion for univariate predictors ($h_k = x^k$) or trigonometric expansion using $h_k = sin(x)$. *General Linear Models* (GLM) aim at fixing the issue mentioned

[1] Also known as **OLS** - ordinary least square.

above which the linear regression model presents, that is, a better fitting relationship between predictors and expected responses. GLMs rely on the assumption that the responses $\{Y_1, \ldots, Y_n\}$ are conditionally independent given the predictors $\{X_1, \ldots, X_n\}$. In addition, it relies on the link function g being differentiable, defined as:

$$g(\mathbb{E}[Y_i|X_i]) = X_i^T\beta. \tag{3.6}$$

3.1.2 Logistic Regression

A logistic regression model is a GLM using a logistic link function, as this model is used for two-class (binary) problems. Consider a general linear model, similar to Equation 3.3 with predictors $\{X_1, \ldots, X_n\}$ and predictor coefficients $\{\beta_0, \ldots, \beta_p\}$. For a two-class data representation, the most common model used for posterior probabilities are (similar to 3.2):

$$\begin{aligned}\mathbb{P}[\mathbb{G} = 0|X = x] &= \frac{exp(\beta_0 + \beta^T x)}{1 + exp(\beta_0 + \beta^T x)},\\ \mathbb{P}[\mathbb{G} = 1|X = x] &= \frac{1}{1 + exp(\beta_0 + \beta^T x)},\end{aligned} \tag{3.7}$$

where $\mathbb{P}[\mathbb{G} = 0|X = x]$ defines the probability of a failure and $\mathbb{P}[\mathbb{G} = 1|X = x]$ defines the probability of a success, given predictor values of x. Using a *logit* transformation; or function, which is the quantile function associated with the logistic distribution, defined as:

$$logit(p) = log\frac{p}{1-p} \tag{3.8}$$

to the above two equations one can see that:

$$log\frac{\mathbb{P}[\mathbb{G} = 0|X = x]}{\mathbb{P}[\mathbb{G} = 1|X = x]} = \beta_0 + \beta^T x. \tag{3.9}$$

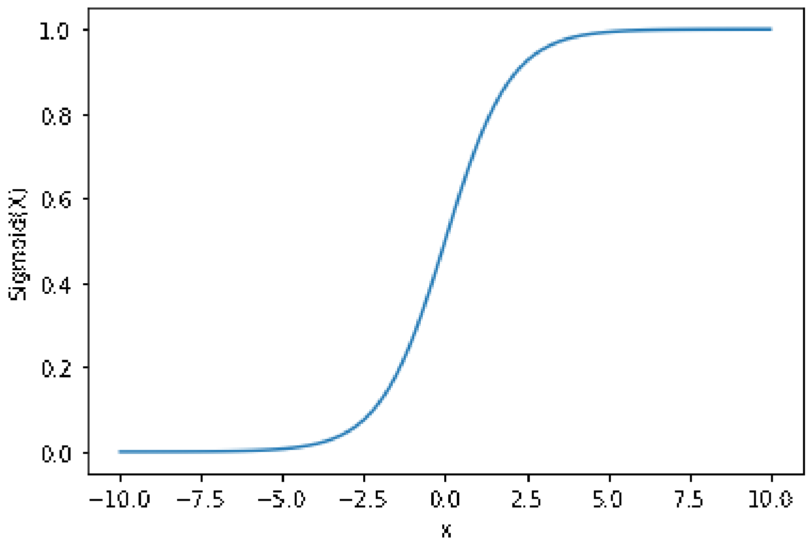

Figure 3.1 – Sigmoid function used as the link function for a logistic regression.

The decision boundary is the set of points for which the log-odds are zero, as defined by a hyperplane $\{x|\beta_0 + \beta^T x = 0\}$[17]. There are at least two popular methods which use logits: Linear Discriminant Analysis and Linear Logistic Regression, with this research paper focusing on the latter. Figure 3.1 illustrates the link function which is used to map the linear combination between the predictors to a probability of landing in each given class (for this paper being Successful/Unsuccessful throw-in). Logistic Regression models are typically fit using maximum likelihood methods, using the conditional likelihood of G (a given class) given X [17]. Fitting the model using the conventional approach based on maximum likelihood relies on two assumptions. First, that all data points are independent but not necessarily identically distributed since predictor X distributions might differ across observations. Second, given predictor X_i, the value of the corresponding response Y_i for observation i, conditional on predictors, is assumed to be Bernoulli distributed with parameter p. Refer to the following equation below of the probability mass function of this Bernoulli distribution:

$$f(g_i;p) = \begin{cases} p, & \text{if } g_i = 1 \\ 1 \text{ - } p, & \text{if } g_i = 0 \end{cases}$$

where g_i is the $i-th$ observation. Consider $\mathbb{P}[\mathbb{G}=0|X=x]=1-p$ and $\mathbb{P}[\mathbb{G}=1|X=x]=p$ from Equation 3.7, the log-likelihood for N observations is:

$$l(\theta)=\sum_{i=1}^{N} log\ p_{g_i}(x_i;\beta). \tag{3.10}$$

The above maximum likelihood equation can be solved using the Newton–Raphson algorithm [17] to estimate our set of predictor coefficients $\{\beta_0,\ldots,\beta_p\}$.

3.1.3 Random Forest Model

Random forests are a common ensemble machine learning method which combines binary regression decision trees to create predictions. Random forests are a combination of tree predictors such that each tree will differ from each other as to stochasticity in their construction, with further details provided below.

Consider a binary classification problem with a discrete response Y with predictors $\{X_1,X_2\}$, each taking values in the unit interval. The top left panel of Figure 3.2 shows a partition of the feature space by lines that are parallel to the coordinate axes. In each partition element, Y is modeled with a different constant. Recursive binary partitions, as per the image on the top right of Figure 3.2, beginning with a split into two regions, and then modelling the response by the mean of Y in each region. The variable and split-point is chosen to achieve the best fit. Then one or both regions are split into two more regions, and this process is continued until some stopping rule is applied [17]. In the top right of Figure 3.2, this process occurred five times, occurring into five regions $R_1,R_2,\ldots R_5$. The regression model which predicts the response Y with some constant, c_m in region R_m:

$$\hat{f}(X)=\sum_{m=1}^{5} c_m I\{(X_1,X_2)\in R_m\}. \tag{3.11}$$

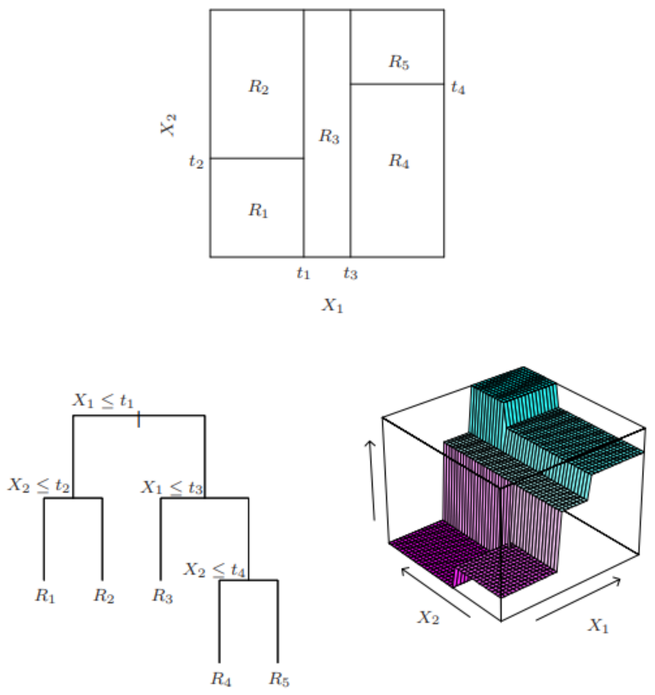

Figure 3.2 – Partitions of a two-dimensional feature space by recursive binary splitting. Taken from [17].

where I is the indicator function. This algorithm of the model can be visualized by the binary tree in the bottom left of Figure 3.2. The bottom right of Figure 3.2 is a perspective plot of the regression surface from this model. The structure of our decision (regression) tree follows this ideology.

The classification decision tree algorithm needs to automatically decide on the splitting variables and corresponding split points. Consider an extension of the above Equation 3.11, such that there is a partition into M regions, $R_1, R_2, \ldots R_M$, and the response is modelled by a constant c_m in each region:

$$T(x) = f(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\} \tag{3.12}$$

Each tree is also defined with the above equation. The optimal $\hat{c}_m$ is the average of the corresponding responses y_i in the given region:

$$\hat{c}_m = ave(y_i | x_i \in R_m) \tag{3.13}$$

For this analysis, a binary classification setting is used in which the responses are either 0 or 1 and the average represents the proportion of successes. Computing and finding the best binary partition is difficult and therefore a greedy algorithm is used to compute the splits for the nodes of the tree. Recursively splitting is applied on the observations resulting into two regions on a given variable x_j. The equation below defines the splitting variable j, the split point s, and the pair of half-planes:

$$R_1(j, s) = \{X | X_j \leq s\}, R_2(j, s) = \{X | X_j > s\} \tag{3.14}$$

As this analysis is given for a classification context, the variables j and s are selected as those that yield that smallest value for the *Gini* Impurity Index, defined below:

$$Gini = \sum_m \pi_m \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{3.15}$$

The *Gini* Impurity Index is a measure of total variance across the K classes. $\hat{p}_{mk}$ represents the proportion of training observations in the m-th region that are from the k-th class (correctly classified) and and $\sum_m \pi_m$ is the proportion of observations sorted into region m. The Gini index takes on a small value if all the proportions are close to zero or one - which is why it is referred to as a measure of node purity [17]. Given that this analysis involves looking at a binary setting, only two classes are considered for Equation 3.15 (K = 2).

This creates the framework for the random forest algorithm, an aggregation of binary decision trees to form a single learner (classifier) by averaging the majority vote of the individual trees. Random forests use a technique to reduce correlation between each of the individual trees. Having many similar decision trees creates predictions that will be highly correlated. Random forests aim at correcting this problem with the *decorrelation* of its predictions.

The algorithm process is as follows: for each tree in the forest, a (bootstrapped) sample is chosen. At each node of the tree, instead of examining all possible feature-splits, a subset of the features $f \subseteq F$ are randomly selected such that F is the set of all possible features. The node then splits on the best feature in f (as opposed to F). The algorithm is defined below.

Algorithm 1 Random Forest Algorithm

```
Require: $n$, Number of predictors
Require: $N$, Number of trees in forest
Require: $S = (x_1, y_1), \ldots, (x_n, y_n)$, the training set.
  for i = 1, ..., N do
    $S^i \leftarrow$ A bootstrap sample from S
    $h_i \leftarrow$ Grow (learn) tree $h_i$
    $H \leftarrow H \cup h_i$
  end for
  Output the ensemble of trees $\{h_i\}_1^N$
  To make a prediction at a new point x:
  return $\hat{f}_{rf}^{M}(x) = \frac{1}{N} \sum_{i=1}^{N} h_i(x)$
```

Growing the tree (in the for loop) for Algorithm 1 means recursively repeating the following steps for each terminal node of the tree, until the minimum node size is reached.

1. Randomly select some subset of the n features.
2. Pick the best variable/split-point among this set.
3. Split the node into two daughter nodes.

3.1.4 Boosted Models

Boosting is another powerful tool in statistics which extends to both classification and regression problems. The general concept of *boosting* is to mix (or combine) classifiers with the goal of creating a strong bond or committee to create better predictions. For this research paper it will be explained using a boosted decision tree. Boosting decision trees works by sequentially growing each tree, each grown using information from the previously grown trees. Boosting does not involve bootstrap sampling. Instead, each tree is fit on previous residuals which is model errors from the previous iteration of the data set [17]. Given a current model (in this case, the singular decision tree), a decision tree is fit to the residuals from the model. This new decision tree is added into the function to update our residuals, by fitting subsequent trees to the residuals to improve our overall classifier. All subsequent trees are trained following this pattern and our final predictions are made using an aggregation of all the trees [17]. Two methods will be described in this section, the forward stage-wise boosting and gradient boosting. The forward stagewise boosting (in a regression context) algorithm below further illustrates how it works.

Algorithm 2 Forward Stagewise Boosting (Regression Tree) Algorithm

```
Require: $X$, Feature matrix
Require: $N$, Number of trees in forest
Require: $d$, Maximum number of splits for each tree (maximum depth)
Require: $y$, Vector of target values
  Define: $\hat{f}(x) = 0$
  Define: $r_a = y_a$ for all $a$ in the training set
  for i = 1, . . . , N do
    Fit a tree $\hat{f}^i$ with at most $d$ splits, to the training data, $(X, r)$
    Update $\hat{f}$ by adding in a shrunken version of the new tree:
    $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^i(x)$
    Update the residuals:
    $r_a \leftarrow r_a - \lambda \hat{f}^i(x)$
  end for
  Output the boosted model prediction:
  return $\hat{f}(x) = \sum_{i=1}^{N} \lambda \hat{f}^i(x)$
```

As mentioned previously for Equation 3.12, a constant c_m is assigned to each node $(R_m, m = 1, \dots M)$ of the decision tree and the predictive rule is:

$$x \in R_m \rightarrow f(x) = c_m \tag{3.16}$$

Following Equation 3.12 with a slight modification, each tree of a boosted decision tree can be defined as:

$$T(x;\Theta) = \sum_{m=1}^{M} c_m I(x \in R_m) \tag{3.17}$$

where $\Theta_m = \{R_m, c_m\}$ is the set of hyper-parameters for region m. Looking at each region such that $\Theta = \{R_m, c_m\}_{m=1}^{M}$, the optimal $\hat{\Theta}$, is found using the Area Under Curve score (AUC and, explained in Section 3.2.1). The parameters are found by minimizing the empirical risk, looked at each region in R_m:

$$\hat{\Theta} \in \underset{\Theta}{\operatorname{argmin}} \sum_{m=1}^{M} \sum_{x_a \in R_m} L(y_a, c_m) \tag{3.18}$$

where the loss function, L is defined as logit log-loss defined above, Equation 3.8. Using the above equations, a boosted tree model can be defined as a sum of such trees:

$$f_N(x) = \sum_{i=1}^{N} T(x;\Theta_i) \tag{3.19}$$

and for each stage (i) of the sequential Algorithm 2, the optimal hyper-parameters Θ_i are solved:

$$\hat{\Theta}_i \in \underset{\Theta_i}{\operatorname{argmin}} \sum_{a=1}^{M} L(y_a, f_{i-1}(x_a) + T(x_a;\Theta_i)) \tag{3.20}$$

for the region set and constants $\Theta_i = \{R_m, c_m\}_1^M$ of the next tree, given the current model (from Equation 3.19) $f_{i-1}(x)$. $T(x_a;\Theta_i)$ is defined as the classification tree from Equation 3.17. The optimal constants (set of parameters) c_m in each region is defined as:

$$\hat{c}_m \in \underset{c}{\operatorname{argmin}} \sum_{x_a \in R_m} L(y_a, f_{i-1}(x_a) + c) \tag{3.21}$$

The numerical optimization is the sum of component vectors - with each data point being a value of the approximating function:

$$f_N = \sum_{i=0}^{N} h_i, h_i \in \mathbb{R}^M \tag{3.22}$$

where h_i is defined as the initial set of hyper parameters:

$$h_i = T(x; \Theta_i) \tag{3.23}$$

beginning with $f_0 = h_0$ as an initial guess and sequentially updating the approximating function based on the sum of the previously induced updates. This method of stage-wise boosting (Algorithm 2) is a greedy strategy. The algorithm at each step looks for the solution tree in which creates the largest reduction in Equation 3.20, given the current model f_{i-1}, it fits $f_{i-1}(x_a)$.

An alternative method is to use the negative gradient (gradient boosting) approach. Consider a tree $T(x; \Theta_i)$ at the $i-th$ iteration, whose predictions are closest to the negative gradient of the loss function below, Equation 3.24. For classification the loss function is the multinomial deviance, however for binary classification (K = 2), simplifies to binomial deviance (cross-entropy) [17]. The binomial deviance is given by:

$$L(y, f(x)) = -\sum_{k=1}^{K} I(y = G_k) log(p_k(x)) \tag{3.24}$$

such that probabilities are produced by the logistic model:

$$p_k(x) = \frac{e^{f_k(x)}}{\sum_{k=1}^{K} e^{f_k(x)}} \tag{3.25}$$

As this is a binary classification task (K=2), the above equation penalizes incorrect predictions only linearly in their degree of incorrectness and furthermore only one tree is needed at each iteration. The tree T_k is fit to its respective negative gradient vector g_k, given by:

$$-g_k = -[\frac{\partial L(y_a, f_k(x_a))}{\partial f_k(x_a)}]_{f(x_a)=f_{i-1}(x_a)} \tag{3.26}$$

This new equation will replace the previously described Equation 3.20. The algorithm below illustrates the gradient boosting algorithm which has just been described. It can be modified for various loss criterion's. Algorithm 2 only uses three tuning parameters: N, d and λ.

Algorithm 3 Gradient Boosting Algorithm

```
Require: $X$, Feature matrix
Require: $N$, Number of trees in forest
Require: $M$, Size of each tree in forest
Require: $y$, Target value of residual values
Require: $k$, Classes Evaluated (2)
Ensure: $f_0(x) = \text{argmin}_c \sum_{a=1}^{M} L(y_a, c)$
  for i = 1, . . . , N do
    for k = 1, 2 do
      for a = 1, . . . , M do
        Compute $r_{aik} = -[\frac{\partial L(y_a, f_k(x_a))}{\partial f_k(x_a)}]_{f=f_{i-1}}$
      end for
      Fit a tree to the computed target $r_{aik}$ creating the regions $R_m$; m = 1, . . . , $M$
      for m = 1, . . . , $M$ do
        Compute $c_m = \text{argmin}_c \sum_{x_a \in R_m} L(y_a, f_{k,i-1}(x_a) + c)$
      end for
      Update $f_{k,i}(x) = f_{k,i-1}(x) + \sum_{m=1}^{M} c_m I(x \in R_m)$
    end for
  end for
  Output the overall boosted model, 2 different (coupled) tree expansion summed over all trees:
  return $\hat{f}(x) = f_{kN}(x), k = 1, 2$
```

Algorithm 3 includes one more, M, the size of each tree within the forest.

There are many more parameters (defined as hyper-parameters) which can be explored of which their results are examined in further sections below. Hyper-parameter tuning was performed using a grid search, designed by a set of fixed parameter values which are essential in providing optimal accuracy based on n - fold cross-validation. The model is trained on each combination of hyper-parameters, selecting the combination which achieves the best

Hyper-parameter	**Description**
Learning rate	Also known as η. After each boosting step, obtained directly from the weights of new features, and η shrinks the feature weights to make the boosting process more conservative.
Maximum Depth	The maximum depth of a tree.
Minimum child weight	Minimum number of samples that a node can represent to be split further.
Subsample	Denotes the fraction of observations to be randomly sampled for each tree.
Column sample by tree	The fraction of features (randomly selected) that will be used to train each tree.
Number of estimators	The number of trees in the model.
Gamma	The minimum loss reduction required to make a split.

Table 3.1 – Hyper-parameters used for the boosted decision tree model.

validation score. All the hyper-parameters used along with their descriptions are shown below in Table 3.1.

3.2 Model Evaluation

3.2.1 Area Under the Curve

For model evaluation various metrics and techniques were used. The area under the receiver operating characteristic curve (AUROC)[2] was used as the main metric due to its universal use. The Area Under the Curve (AUC) score is a method to quantify the discriminatory performance of a two-class classifier [18], which for this research paper is whether a throw-in is *Successful* or *Unsuccessful*. The ROC graph is created by plotting the true positive

[2]Also known as the *ROC curve* or *AUC score*.

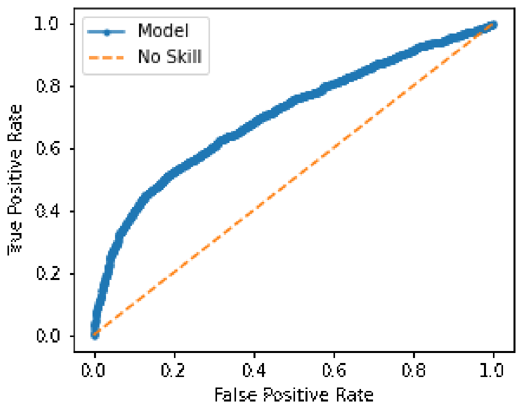

Figure 3.3 – ROC Curve.

rate against the false positive rate, using various classification thresholds. A score of 0.5 for the AUC signifies no predictive power from the model (essentially guessing) whereas a score of 1 signifies the classifier fully discriminates between successes and failures. Figure 3.3 illustrates a ROC curve example showing models with a score of 0.5 with the dotted line and a models score between 0 and 1. The performance can be further looked at using a Confusion Matrix, which is used as a method to visualize the classifiers performance. The matrix looks at how often the two classes are being correctly classified by looking at true/false positives and true/false negatives. Figure 3.4 illustrates a confusion matrix. To create the confusion matrix, a probability threshold is fixed for classification. The example uses positive/negative as the two cases, whereas for this paper the two corresponding classes are a Successful/Unsuccessful throw-ins.

There are many performance metrics which can be taken from the confusion matrix, but the two most commonly used and analysed are the *precision* and *recall*:

$$precision = \frac{TP}{TP + FN} \tag{3.27}$$

$$recall = \frac{TP}{TP + FP} \tag{3.28}$$

$$\mathbb{F} = 2 * \frac{precision * recall}{precision + recall} \tag{3.29}$$

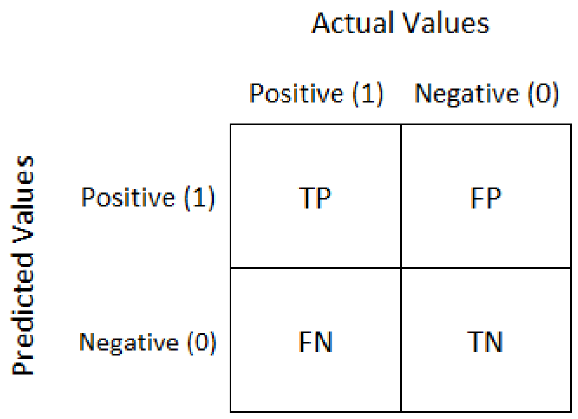

Figure 3.4 – Confusion Matrix. Taken from [19]

where a true positive (TP) is when the model correctly predicts the positive class and, a true negative (TN) is an outcome where the model correctly predicts the negative class. A false positive (FP) is an outcome when the model incorrectly predicts the positive class and, a false negative (FN) is an outcome when the model incorrectly predicts the negative class.

Precision, Recall and Specificity (True Negative Rate) metrics illustrate other facets of performance than just the AUC score. Specifically, measuring accuracy instead of discriminatory power which is important in the results section due to the imbalanced nature of the data. As specificity is the true negative rate, it will help show how well Successful/Unsuccessful are identified, which is important as there is close to a five to one Successful/Unsuccessful ratio. In addition, there is interest in the *recall* values as it measures what percentage of Unsuccessful throw-ins can be detected. In other words, given an Unsuccessful throw, can the model developed correctly determine that it is Unsuccessful. The F1-Score 3.29 is a measure that combines the previously mentioned precision and recall and is used a measure of a test's accuracy.

3.2.2 Shapley Additive Explanations

To better understand the importance of the various features within the models, Shapley values were calculated. SHAP assigns each feature an importance value for a particular prediction. This metric provides a perspective on variable importance that is different from the traditional feature importance (F-score). Traditional feature importance is based on the decrease in model performance, where SHAP is based on the magnitude of feature attributions.

Suppose there is a linear model prediction similar to that of Equation 3.3. The contribution ϕ_j of the *j-th* feature on the prediction $\hat{f}(x)$ is:

$$\phi_j(\hat{f}) = \beta_j x_j - \mathbb{E}(\beta_j X_j) = \beta_i x_j - \beta_j \mathbb{E}(X_j) \tag{3.30}$$

Where $\mathbb{E}(\beta_j X_j)$ is the mean effect estimate for feature j. The contribution is the difference between the feature affect minus the expected affect - giving us an estimate of how much each feature contributed to the prediction. Summing up all the feature contributions for one instance results in the following:

$$\sum_{j=1}^{p} \phi_j(\hat{f}) = \sum_{j=1}^{p} (\beta_j x_j - \mathbb{E}(\beta_j X_j)) = (\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - (\beta_0 + \sum_{j=1}^{p} \mathbb{E}(\beta_j X_j)) = \hat{f}(x) - \mathbb{E}(\hat{f}(X)) \tag{3.31}$$

The above formula shows the predicted value for the data point x_j minus the mean predicted value. This creates the framework for the Shapley ideology. The Shapley value of a particular feature of your model is its contribution to the prediction, weighted and summed over all possible feature value combinations[3]. Define $\phi_j(\pi)$ as the Shapley value of a particular

[3]https://christophm.github.io/interpretable-ml-book/shap.html

feature j. Define π as the prediction function mapping predictors to a predicted value for features in S, where S is a subset of the features used in the model and p is the number of features.

$$\phi_j(\pi) = \sum_{S \subseteq \{1,\ldots,p\} \setminus j} \frac{|S|!(p - |S| - 1)!}{p!} (\pi(S \cup \{j\}) - \pi(S)) \tag{3.32}$$

To estimate the Shapley value, all possible combinations of feature possibilities must be explored. The value of an individual feature value is the average change in the prediction that the current combination room receives when the feature value joins.

Strumbelj and Kononenko [20] propose an approximation (to estimating the Shapley value) using Monte-Carlo sampling. The approximation algorithm begins by selecting a data point x, a feature j and the number of iterations M. At each iteration we sample a random data point z. Using observations x and z, two new data points are created, x_{-j} and x_{+j}. The new data point x_{+j} is the combination of x and z (randomly ordered) with the j-th entry being substituted by the value x_j (taken from x). Similarly, the new data point x_{-j} is the combination of x and z (randomly ordered) with the j-th entry being substituted by the value z_j (taken from z). The difference of the prediction of these two data sets is computed for this m-th iteration and defined as:

$$\hat{\phi_j^m} = (\hat{f}(x_{+j}^m) + \hat{f}(x_{-j}^m)) \tag{3.33}$$

looking at every difference, summing through each iteration and taking the average is defined as:

$$\hat{\phi_j}(x) = \frac{1}{M} (\hat{f}(x_{+j}^m) + \hat{f}(x_{-j}^m)) \tag{3.34}$$

Repeating this procedure for each feature is completed to get each of their individual Shapley values. The algorithm is shown explicitly below (Algorithm 4).

Algorithm 4 Approximate Shapley estimation for single feature value.

```
Require: $M$, Number of iterations
Require: $x$, Instance of interest
Require: $j$, Feature index
Require: $X$, Data matrix
Require: $\hat{f}$, Machine learning prediction model
  for m = 1, ..., M do
    Draw (randomly) z from X
    Choose a random permutation $\mathcal{O}$ of the feature values
    $x_0 \leftarrow (x_{(1)}, \ldots, x_{(j)}, \ldots, x_{(p)})$
    $z_0 \leftarrow (z_{(1)}, \ldots, z_{(j)}, \ldots, z_{(p)})$
    $x_{+j} \leftarrow (x_{(1)}, \ldots, x_{(j-1)}, x_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
    $x_{-j} \leftarrow (x_{(1)}, \ldots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
    $\Delta_m(x_j) \leftarrow \hat{f}(x_{+j}) - \hat{f}(x_{-j})$
  end for
  return $\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \Delta_j^m(x_j)$
```

Chapter 4

Results

4.1 Data Visualization

Before performing any analyses, it is important to obtain a better understanding of the data set and the features that were created. A box plot is a commonly used method of displaying the distribution of data based on five metrics: minimum, maximum, median, first and third quartiles. A box is shown which is defined as the Inter Quartile Range (IQR) - spanning from the first and third quartiles. A Box plot for each of the features described in Section 2.2.1 are shown in Section A.3 of the Appendix.

The density plots shown are also useful as they help illustrate the distribution of the features, which will power the models developed. The density plots (along with the box plots) have been split to show various scenarios: the complete data set, Successful/Unsuccessful throw-ins for the *first touch* definition and Successful/Unsuccessful throw-ins for our *threshold retention* definition. Within each of the box plots are a small triangle which represents the mean value for the particular set of throw-ins. This inclusion to the plot is meaningful as it serves as a quick way to see if the data set in question is creating the inference one would

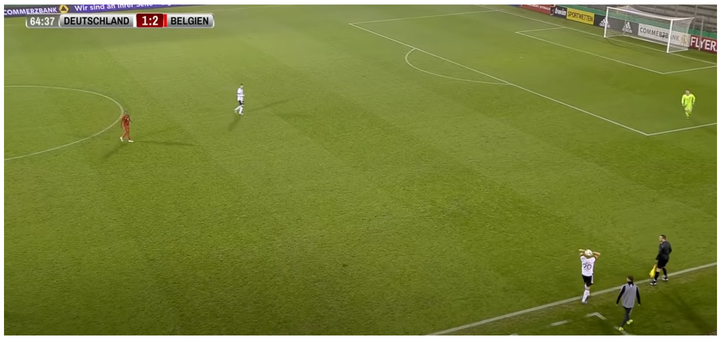

Figure 4.1 – High angle throw illustration. Angle to the receiver measured at 141.19° with a *Pressure* value of 0. Taken from a Germany vs. Belgium U-21 European Qualifiers game, played on 17/11/2019.

assume. For example, for the *Pressure* feature (Figure A.8), all the unsuccessful throws have a lower mean value in comparison to the Successful throw-ins. Although this was assumed, the logic is well-founded as the lower the amount of pressure on a potential receiver, the more likely it is completed. Some other assumptions that were made when creating the features were also confirmed. The angle feature as explained in Section 2 is calculated on a 0° to 180° scale, normalized from the thrower to the potential receiver with the throwing team always playing from left to right. Given the nature of this feature, the throw-ins with a low angle are forward throw-ins and the throw-ins with a high angle are backward throw-ins.

Low angle throw-ins (Figure 4.2) are generally being thrown to receivers which have defenders in close proximity[1] and thus it can likewise be assumed that given this high pressure, the throw-in completion rate is low. On the contrary, high angle throw-ins (Figure 4.1) are generally being thrown to open receivers (little to no pressure) and thus would expect a high throw-in completion rate.

Figure A.13 illustrates the box plots for angles and it can be seen that the mean value for the Successful throw-ins is higher than the mean value for the Unsuccessful throws, confirming our initial beliefs. Figures 4.1 and 4.2 show examples of these two types of throw-ins. Figure 4.1 is a Successful throw with an angle to the receiver of 141.19° and one can easily see it is

[1]As you are throwing in the direction of the opposing team.

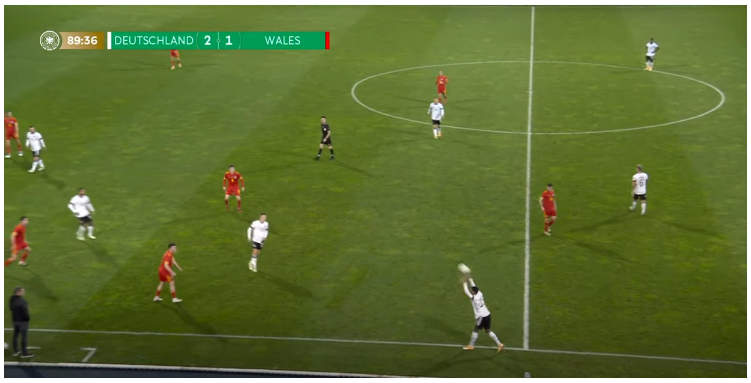

Figure 4.2 – Low angle throw illustration. Angle to the receiver measured at 33.09° with a *Pressure* value of 0.531. Taken from a Germany vs. Wales U-21 European Qualifiers game, played on 17/11/2020.

completed with no defenders in sight (and correspondingly a *Pressure* value of 0). On the other hand, Figure 4.2 shows an Unsuccessful throw with an angle to the receiver of 33.09° and corresponding *Pressure* value of 0.531. This picture illustrates what a vast majority of throw-ins with a low angle value look like, especially in the opposing team's side of the field.

The *Time2Throw* variable is another feature that when looking at the box plot, one can have a high level of confidence in a hypothesis of how it would react in terms of Successful and Unsuccessful throw-ins. Naturally, it can be assumed that faster throw-ins would lead to greater success as the faster a throw-in is taken, the less time the opposing team has to create coverage and defend the throw properly. Looking at Figure A.11, it can be seen that Successful throw-ins have a mean time of around 13 seconds, while Unsuccessful throw-ins have a significant increase in mean time to approximately 16-17 seconds.

The *Pressure* feature showed the biggest discrepancy in the mean value of the throw-in data set between Successful and Unsuccessful throw-ins. This confirmed the prior hypothesis mentioned in Section 2.2.3 of lower *Pressure* equates to higher success rates and higher *Pressure* equates to lower success rates. Figure A.8 provides mean *Pressure* values for Successful throw-ins of approximately 0.4, while the Unsuccessful throw-ins provide mean *Pressure* values of 0.78 and 1.03.

The *Distance* feature however provided some intriguing results which will be discussed further when implementing this feature into the models mentioned in Section 3. One would naturally assume that shorter throw-ins lead to higher success as the ball must travel a shorter distance and therefore less things can go wrong. However, looking at Figure A.6, it can be seen that this is the case for the *first touch* definition but not for the *threshold* definition. This can be explained by the two definitions of Success; *first touch* success just needs to *hit* the receiver to be Successful while the *threshold* success needs a player to maintain possession (which is inherently much more difficult). Not only is it more difficult to do, but also much can occur within 7 seconds. Effectively, one can not predict what will happen 7 seconds into the future but, immediately into the future (which *first touch* success looks at), reasonable predictions and assumptions can be made.

Another way to get a better understanding of the data set is to look at the correlation values between all features along with the two responses. Interest lies on what features impact throw-in Success along with which features interact with each other (and to what extent they do, if any at all).

With values of -0.38 and -0.29 for the *Pressure* correlation to *first touch* success and *threshold* success respectively, the hypothesis can be confirmed that lower *Pressure* on the receiver leads to higher success rates. Another interesting observation from Figure 4.3 is the *Distance* feature, with negative correlation to *first touch* success but a slightly positive correlation to *threshold* success. This will be explored further below when powering our model with the features and looking at practical applications but the initial assumptions as to why this is occurring is the same reason mentioned above. The reasoning being that it is hard to predict what will occur 7 seconds in the future (along with all other possible events occurring during that period), but predicting the immediate future is more feasible.

Another interesting takeaway from Figure 4.3 is the *Time2Throw* feature, which has a neg-

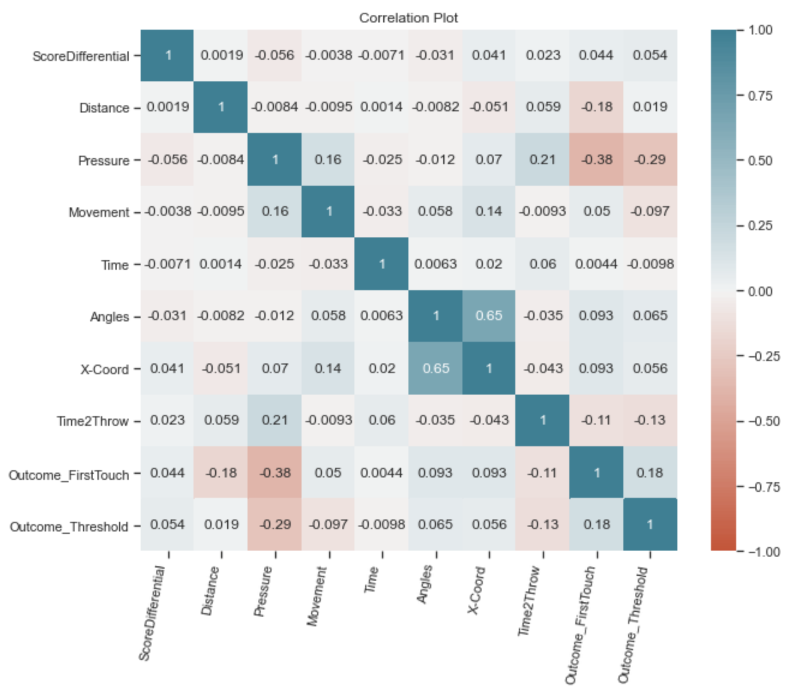

Figure 4.3 – Correlation plot of all the features described in Chapter 2 along with the two response variables, *first touch* and *possession* success definitions.

ative correlation value to both of the success definitions. This directly matches what was previously discussed for this feature with its box plot (Figure A.11) mean values being lower for Successful throw-ins versus higher mean values for Unsuccessful throw-ins.

The last feature considered is the *movement* feature. Intuitively, it would be fair to think that more movement leads to greater success, as the goal of creating movement during a throw-in is to reduce the *Pressure* being received from the defender to receive the ball. When looking at the box plot for *movement*, the distribution is very similar for all of the scenarios along with the mean values of each of the data sets. Similarly, on the correlation plot, Figure 4.3, the correlation values for the two success definitions are 0.05 and -0.097 respectively.

These values are surprising, as it can be considered intuitive that they would provide a greater impact. This feature and possible adjustments to the implementations of it will be further discussed below with the goal of adjusting it to create more value.

The feature is engineered to be forward looking as it looks at a certain point in the future (the time of reception) and calculates how much distance the receiver will cover to the time of the throw-in. It is possible that throughout that time there is an instance in which the throw-in can be completed with greater success. This feature has a great deal of potential and will be explored further in future work, with the main goal of looking at the *movement* feature values throughout the time of the throw-in and seeing if there are possibly more optimal times to complete the throw-in.

4.2 Model Results

The model results explored in this section will follow the same order as the theory was presented in Chapter 3. All analyses performed for this Chapter are performed on the data set previously mentioned in Chapter 2.2.4, a total of 6119 throw-ins across various Germany National Team soccer games. For all the analyses below, both success definitions will be explored to draw insight from the results. For the three models which will be discussed, a train/validation along with a test set was used. A completely random 80/20 split of the overall data set was made, where the 80 percent was used for training/validation and the final 20 percent was used for testing.

4.2.1 Logistic Regression Results

The analysis looked at all combinations of features to see which combination provided the best results, in terms of highest out-of-sample AUC score, as explained in Section 3.2.1. Each combination of features was trained and then tested implementing 5-fold cross-validation, using a standard logistic regression. The results presented in Tables 4.1 and 4.2, which respectively show in-sample and out-of-sample results, are using the feature combination

Feature	**Coefficient**	**T-Value**	**P-Value**
Distance	0.042/-0.008	9.261/-1.381	0.000/0.167
Angle	0.06/0.013	8.575/13.02	0.000/0.000
Game Time	0.03/0.01	2.877/6.981	0.004/0.000
Score difference	0.06/0.05	2.982/2.047	0.003/0.041
Time to take the throw-in	-0.009/0.004	-3.634/1.199	0.000/0.231
Receiver movement	0.002/0.219	0.212/15.314	0.832/0.000
X-Coordinate	-0.001/-0.005	-0.624/-2.813	0.533/0.005
Pressure	-1/-1.64	-16.126/-20.392	0.000/0.000

Table 4.1 – Logistic Regression Model Coefficients for In-Sample Training. AUC score of 0.701 for *threshold*and 0.830 for *first touch*. Log-likelihood value of **-2952.1** for *threshold* definition and log-likelihood value of -1901.7 for *first touch* success. The **black** values used the 7 second *threshold* definition of success and the red values used the *first touch* definition for success.

which yielded the highest AUC score when looking at all combinations. It was found that the combination of using all eight predictors provided the highest AUC score for both Success definitions using in-sample model training.

Both in-sampling and out-of-sampling results are shown in Tables 4.1 and 4.2 below. Along with the AUC score of these models, the log-likelihood value, the coefficient value, t-value and p-value are also shown. The T-value is the t-statistic which measures the ratio of predicted value of a parameter from its hypothesized value to its standard error. The formula is shown below.

$$t_{\hat{\beta}} = \frac{\hat{\beta} - \beta_0}{S.E.(\hat{\beta})} \tag{4.1}$$

where β_0 is a known value for the parameter, $\hat{\beta}$ is the estimated value of the parameter β and S.E is the standard error for the estimated parameter.

The following coefficients were obtained for the combination of best predictors, which were retained for the final model and used for all further analyses below. The **black** value is the values using the 7 second *threshold* definition of Success and the red value is the *first touch*

Feature	**Coefficient**	**T-Value**	**P-Value**
Distance	0.026/-0.02	3.065/-2.180	0.002/0.03
Angle	0.006/0.014	4.375/6.98	0.000/0.000
Game Time	0.004/0.012	2.336/3.827	0.02/0.000
Score difference	0.001/0.008	2.326/1.37	0.02/0.172
Time to take the throw-in	-0.005/0.001	-0.906/1.69	0.365/0.091
Receiver movement	0.008/0.245	0.438/8.3	0.661/0.000
X-Coordinate	-0.0003/-0.006	-0.081/-1.496	0.935/0.135
Pressure	-0.982/-1.92	-7.95/-10.823	0.000/0.000

Table 4.2 – Logistic Regression Model Coefficients for Out-of-Sample Training. AUC score of 0.697 for *threshold* and 0.821 for *first touch*. Log-likelihood value of **-747.79** for *threshold* definition and log-likelihood value of -446.97 for *first touch* success. The **black** values used the 7 second *threshold* definition of success and the red values used the *first touch* definition for success.

definition for Success.

Similar to the results from the box-plot and density distributions shown above, the predictor coefficients can be observed. For example, the one predictor which allows to make the easiest inference from the coefficient is *Pressure* as its coefficient is always negative for all the scenarios. The p-value for *Pressure* for all the scenarios are also less than 0.05 indicating it is providing significance to the model. The other feature with near 0 p-value for both Success definitions is *angle*. The resulting box plot for this feature (Figure A.13), was expected given the reasoning mentioned in Section 4.1 (in regard to the nature of low or high angles throw-ins).

When looking at p-values of strictly less than 0.05 (not close to 0), features *Distance* and *Game Time* become significant for out-of-sample performance. For in-sample performance, along with *Pressure* and *Angle*, the features *Game Time*, *Score Difference*, *Time to throw-in* were also significant for both Success definitions.

The AUC scores are glaringly different for both of the Success definitions with values for

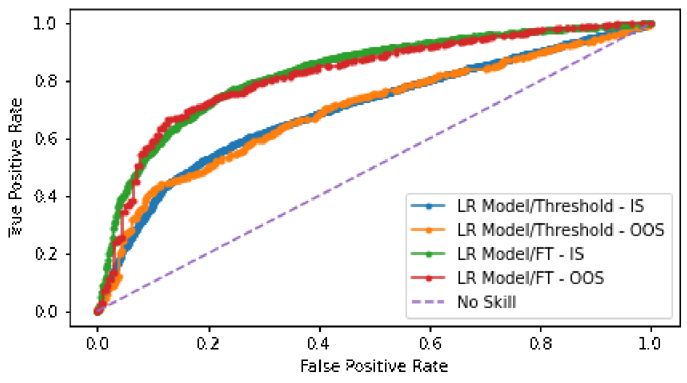

Figure 4.4 – Logistic Regression model ROC curves. The AUC score for *threshold* success in-sample is 0.701. The AUC score for *first touch* success in-sample is 0.830. The AUC score for *threshold* success out-of-sample is 0.697. The AUC score for *first touch* success out-of-sample is 0.821.

the *first touch* success being in the 0.8's and the *threshold* success hovering around 0.7. Intuitively, this leads back to the prior discussion in which most of the features that fed are given at a frozen moment in time, not 7 seconds into the future, with the result being that predictive power of that time is nowhere near as great.

Figure 4.4 displays the ROC curves for both in-sampling and out-of-sampling testing using the logistic regression model. There are several observations to take from Figure 4.4. Both Success definitions are shown on the plot, where it becomes clear the greater predictive power which the *first touch* definition provides.

The most significant outcome from Figure 4.4 is the small discrepancy between in-sampling and out-of-sampling AUC scores. For *threshold* success the discrepancy is 0.04 and for *first touch* success the discrepancy is 0.09. Both the success definitions have a higher in-sample AUC score compared to the out-of-sample score. Given how small the discrepancies are for this model, they can be considered negligible. The more important and encouraging sign is that given these small discrepancies, the results indicate that the model is learning meaningful manner and is then able to apply its knowledge to new unforeseen data. This most likely indicates the absence of substantial overfitting. This generalization can be made

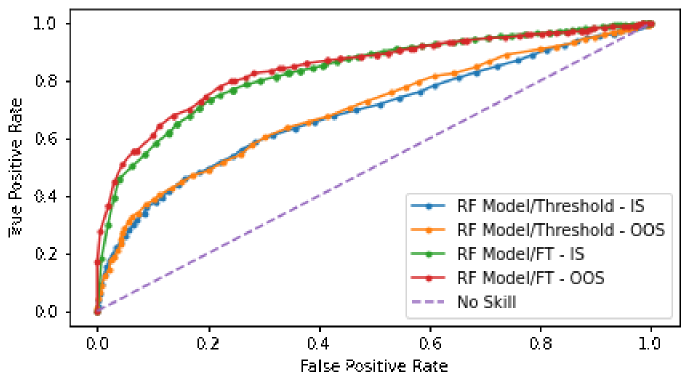

Figure 4.5 – Random Forest model AUROC curves. The AUC score for *threshold* success in-sample is 0.688. The AUC score for *first touch* success in-sample is 0.832. The AUC score for *threshold* success out-of-sample is 0.698. The AUC score for *first touch* success out-of-sample is 0.845.

due to the nature of the data which includes throw-ins from various age groups, leagues and genders.

4.2.2 Random Forest Results

Random forest model results are now explored and discussed following the same sequence as above. Each combination of features was trained and then tested implementing 5-fold cross-validation, using a random forest model. Both out-of-sample and in-sample model results will be shown along with feature variable importance scores and Shapley scores which was described in Section 3.2.2. Figure 4.5 is the ROC curve for both in-sampling and out-of-sampling results using our random forest model. A constant number of trees value of 75 was used for both definitions.

The AUC values for the random forest model follows a similar pattern with the logistic regression model (Figure 4.4) given that the ratio[2] from in-sample and out-of-sample are not drastically different. The actual AUC scores are similar for both Success definitions in the out-of-sample and in-sample testing. Figures 4.7 and 4.6 display the variable importance

[2]Ratio defined as the in-sample over out-of-sample AUC scores.

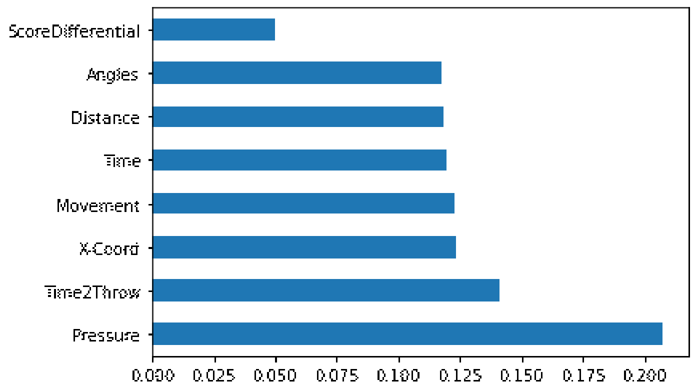

Figure 4.6 – Random Forest Feature Importance Variable for *threshold* success.

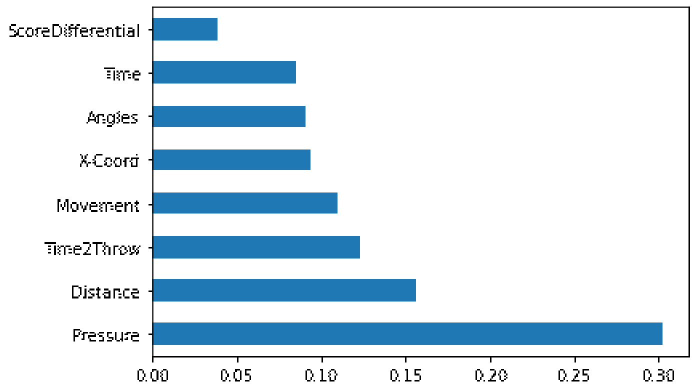

Figure 4.7 – Random Forest Feature Importance Variable for *first touch* success.

using the random forest model with a constant number of trees parameter of 75, for *first touch* success and *threshold* success definitions respectively. Variable importance can be used for both random forests and gradient boosted decision trees. At each split in each tree, the improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees in the forest separately for each variable [17].

The biggest observation from the feature importance plots shown on Figures 4.6 and 4.7 is the impact which the *Pressure* variable provides. For both definitions, it provides the greatest impact, more specifically it was used most often to split a node for the trees. Another interesting observation is the drastic difference for the *Distance* feature for the two Success definitions. For first touch success (Figure 4.7) it ranks as the clear second in importance but for threshold success it is third last in importance (Figure 4.6).

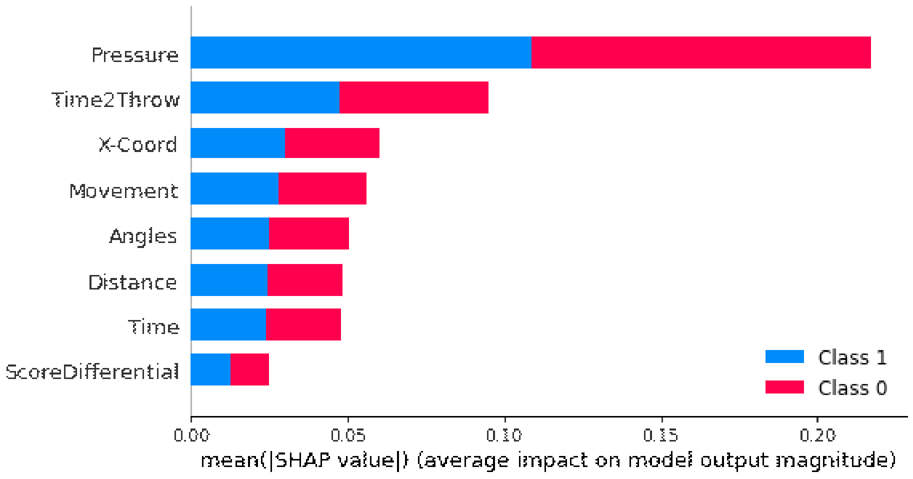

Figure 4.8 – Shapley Values for *threshold* success.

The main deduction from the two feature importance plots is the leading presence which the *Pressure* feature presents to the model. This stems back to the discussion throughout this analysis of the difficulty to develop predictions so far out in the future. If there is little to no pressure while in possession, it is likely that possession will be maintained for a decent time period regardless of the seven other feature values. The *score differential* feature also ranks last for both definitions, which was expected with little correlation to the Success definitions. This will be explored more in depth in the practical applications section once certain throw-ins are isolated. For certain scenarios this feature starts to provide value. Aside from those three feature observations, all the other features ranked were fluid throughout the two definitions as they all ranked close in value and thus hard to extract any meaningful remarks.

Figure 4.8 and 4.9 illustrates the Shapley values for the *threshold* and *first touch* success definitions respectively. Each of the model feature values are further split into the two response variables and how much impact they individually provide. It is taken from Equation 3.31 and looking at both classes' impact on the model, with the two classes being Successful and Unsuccessful throw-ins (Class 1 and Class 0 respectively).

Similar observations to those mentioned for the traditional feature importance figures can

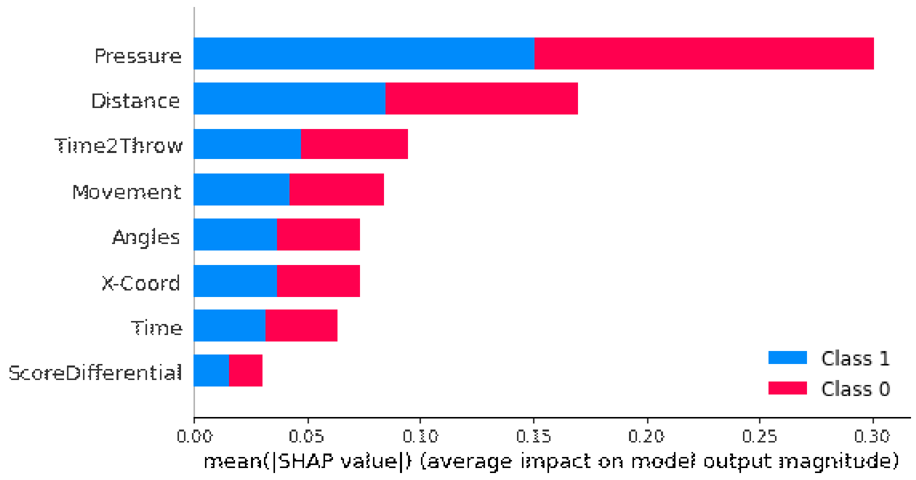

Figure 4.9 – Shapley Values for *first touch* success.

be made for the Shapley feature importance plots. Once again, *Pressure* remains the most valuable feature with approximately an equal impact stemming from each of the two classes. Similar to the traditional feature importance shown above, the *Distance* feature has drastically different rank of importance (or model impact in this case) between the two definitions. The *Time to Throw* feature remains in the top three for both definitions like the previous feature importance figures. It is also relevant and encouraging to see that both classes are affecting the model features at about the same ratio (red to blue on the figures), and none are substantially slanted to one class.

4.2.3 Boosted Decision Tree Regression Model

Boosted decision tree model results are now explored and discussed following the same sequence as above. The model construction was explained in Section 3.1.4, along with the algorithm and hyper-parameters used for these analyses.

Before any parameter tuning was completed, the model was evaluated using ad-hoc hyper-parameters[3] to decide which combination of features to use like the two above methods with

[3]Used a vanilla set of parameters, a number of estimator value of 10 and max tree depth value of 3.

Hyper-parameter	**First-Touch Value**	**Threshold Value**
Learning rate	0.1	0.05
Maximum Depth	5	3
Minimum child weight	10	40
Subsample	0.8	0.9
Column sample by tree	0.8	0.9
Number of estimators	40	35
Gamma	0.08	0.02

Table 4.3 – Optimal hyper-parameter values selected from the grid search for each of the two success definitions.

logistic regression and random forest. The features were chosen from the in-sample tests conduced on the 80 percent of date (train/validation data sets). The combination of all 8 features provided the best AUC score for both success definitions. The AUC scores using the ad-hoc parameters was 0.701 for *threshold* and 0.824 for the *first touch* success.

Once the combination of features was chosen for both definitions, our hyper-parameters (from Table) 3.1 were tuned using a grid search with in-sample AUC score as the evaluation metric (to maintain consistency with the other analyses of this paper). The grid search is done using 5-folds and the parameters which were inputted for the grid search were chosen through various trials of different values and analysing the AUC scores and log-loss curves of the individual hyper-parameters. After many trials and errors with various parameters, 50625 fits were evaluated to choose the optimal hyper-parameter values, one for each of the Success definitions. The set of hyper-parameters chosen from the grid search are illustrated in Table 4.3.

Once the optimal hyper-parameters have been found using the grid search previously described, an out-of-sample model evaluation using the remaining 20% of the data set can be conducted, which has been set aside from the training and validation sets. The ROC graph for both Success definitions is shown in Figure 4.10, along with the AUC scores achieved for

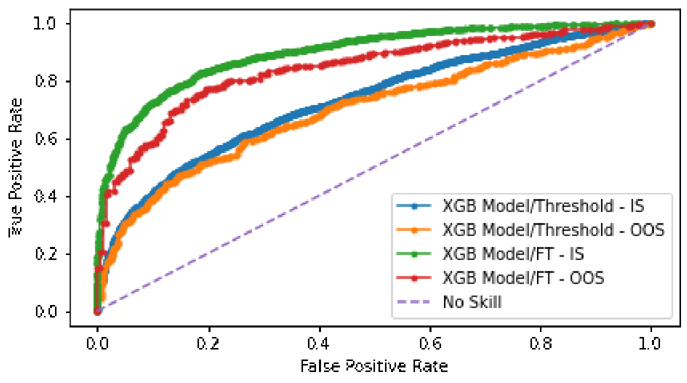

Figure 4.10 – Boosted decision tree model (XGB) AUROC curves. The AUC score for *threshold* success in-sample is 0.731. The AUC score for *first touch* success in-sample is 0.898. The AUC score for *threshold* success out-of-sample is 0.701. The AUC score for *first touch* success out-of-sample is 0.842.

both out-of-sample and in-sample model evaluation.

The AUC scores achieved for each of the two definitions follow the same pattern where higher values were achieved with *first touch* success versus *threshold*. Another similarity is the ratio from out-of-sample to in-sample AUC scores, although the in-sample scores for both definitions are higher, there is not a drastic drop-off to new out-of-sample data. This is an encouraging sign as the AUC scores are higher than logistic regression for both in-sample and out-of-sample definitions and the ratio between the two was maintained, which appears to not be just a scenario of being able to over fit the data.

It has also become quite clear and evident that after testing three different models using these two definitions for success and constantly achieving better scores using *first touch* as supposed to *threshold* that one is more predictive then the other, given the current set of features used for this analysis. It does not mean that using it is necessarily **better**, but given the tests that currently have been done, the *first touch* success is more easily predictable. In the next section, when evaluating practical uses it will be shown that there is a time and place for both to be evaluated and used, thus to conclude it is not as if the *threshold* success provides no value.

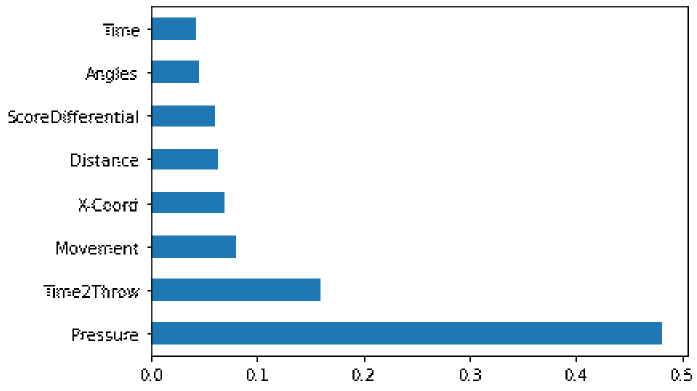

Figure 4.11 – Boosted Decision Tree Feature Importance Variable for *threshold* success with tuned hyper parameters.

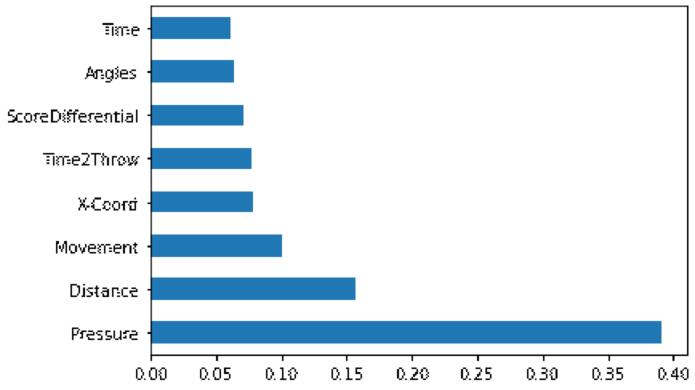

Figure 4.12 – Boosted Decision Tree Feature Importance Variable for *first touch* success with tuned hyper parameters.

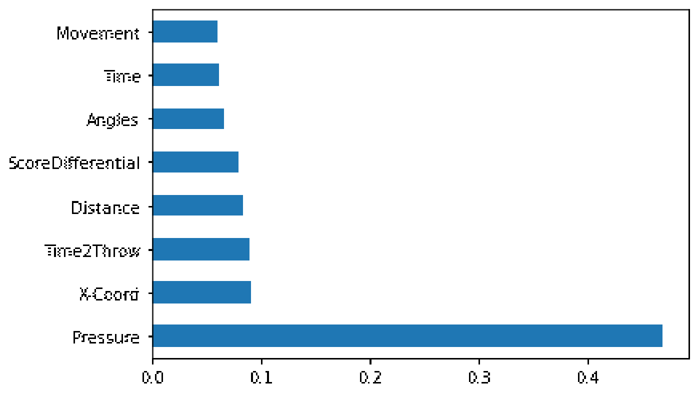

Figure 4.13 – Boosted Decision Tree Feature Importance Variable for *threshold* success with ad-hoc hyper parameters.

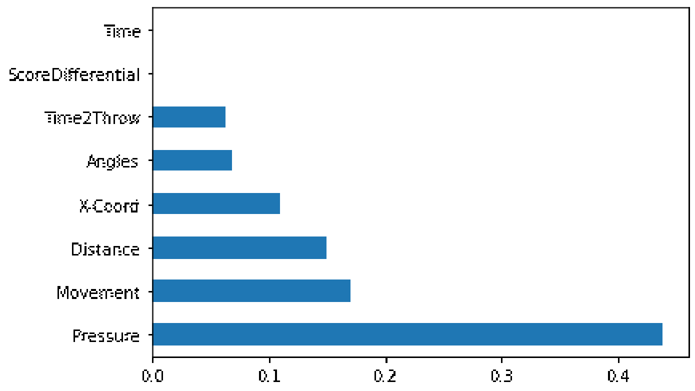

Figure 4.14 – Boosted Decision Tree Feature Importance Variable for *first touch* success with ad-hoc hyper parameters.

Some interesting observations can be made from these figures in terms of inference on the model features and the impact they have made through the variable importance metric. Another interesting remark is that once again the feature *Distance* ranks highly on the *first touch* success (Figure 4.12) but not for *threshold* success (Figure 4.11). This comes back to the recurring theme that it is hard to predict what is happening in the future but logically completing (touching the ball) in the immediate future, a low *Distance* value to the throw is crucial. Figure 4.15 and 4.16 illustrates the mean absolute value of Shapley for the *threshold* and *first touch* success definitions respectively from our tuned boosted decision tree model. Many of the same observations can be made from these values, such as the clarity that *Pressure* is the most impactful model feature. The *Distance* feature again follows the same

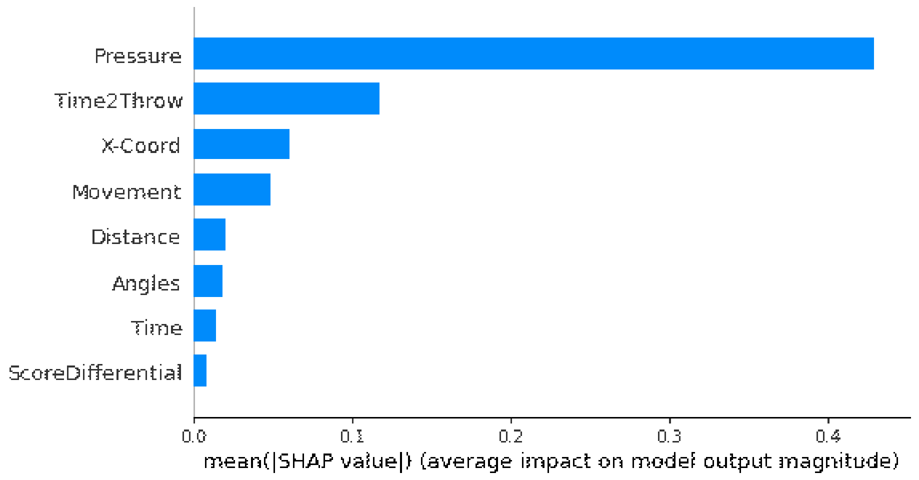

Figure 4.15 – Boosted Decision Tree Shapley Values for *threshold* success with tuned hyper parameters.

pattern of being highly ranked for the *first touch* success but gets lost in the shuffle for the *threshold* success. Another familiar pattern is that the *score differential* model feature is clearly the least impactful throughout all of the models explored.

The reason why Figures 4.15 and 4.16 illustrate just a blue bar as opposed to figures 4.8 and 4.9 separating by class is all due to how the packages are configured. Figures 4.15 and 4.16 which use *XGBoost* are created using predictions which explain the output margin of the trees which is related to the probability of the positive class (Successful class). However, Figures 4.8 and 4.9 use the *RandomForestClassifier* which is treating the binary prediction as predicting two classes (the probability of the positive class and the probability of the negative class).

The Shapley Figures 4.15 and 4.16 can be manipulated to dig further into the feature value on model output. Each data point for each of the feature values can be plotted as a function of their Shapley value (impact on the model output). The Shapley figures which have been previously shown (Figures 4.15 and 4.16) are created by taking the mean value for each of the features from Figures 4.17 and 4.18.

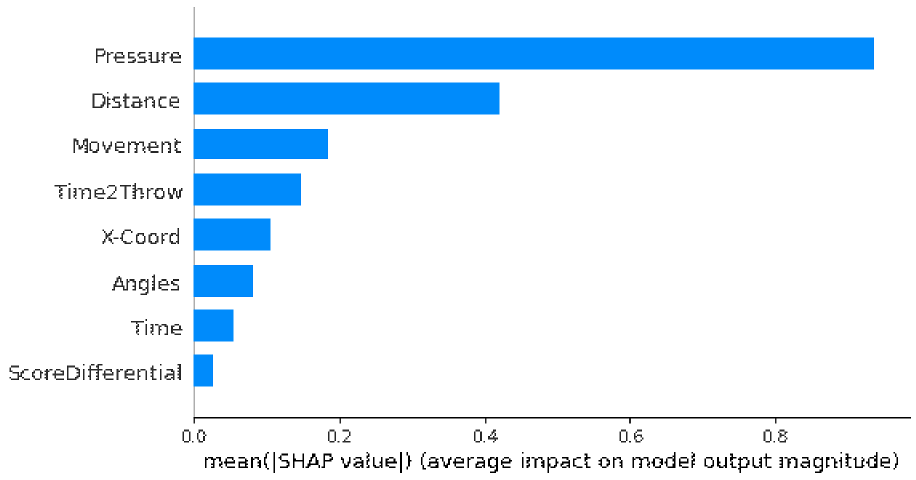

Figure 4.16 – Boosted Decision Tree Shapley Values for *first touch* success with tuned hyper parameters.

Along with the figure, the numerical mean Shapley value is shown for each of the factors (the value which creates the blue bars for Figures 4.15 and 4.16). For example, the Shapley value of 0.41 for the *Pressure* feature for Figure 4.17 is how the first blue bar on Figure 4.15 is created.

The Shapley values calculated for Figures 4.17 and 4.18 are shown rounded up to two significant figures. As a result when dots are shown on the same Shapley value, it means that after being rounded, multiple observations have the same value.

This type of figure is interesting to inspect as one can directly investigate which values of feature values impact the model more and which are redundant. For example, inspecting the *time* feature, some comparisons can be made using these figures which cannot be seen just using the mean Shapley values. The *time* feature for Figure 4.17 has a mean Shapley value of 0.01 and the values seem to cluster around 0 regardless of the feature value. However for the same feature but looking at the *first touch* figure, Figure 4.18, which has a mean Shapley value very close to *threshold* (0.06 vs 0.01) the cluster of feature value is very spread out. Spread out in the sense that the low feature value (blue dots - beginning of the game) seem to provide the positive Shapley value and the high feature value (red dots - ending of

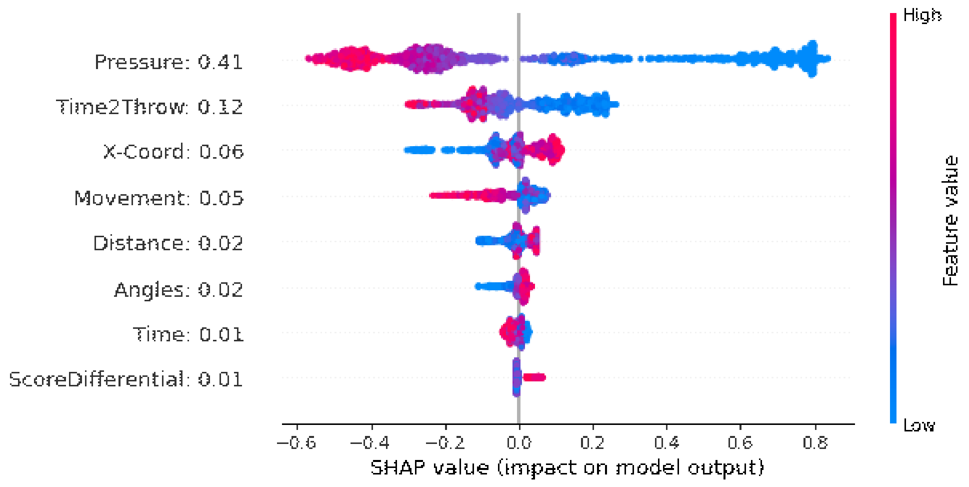

Figure 4.17 – Boosted Decision Tree Shapley Values for *threshold* success with tuned hyper parameters. Shapley value for each prediction is individually plotted as a function of the feature value. Each observation can be deconstructed into their respective model impact using Shapley feature values.

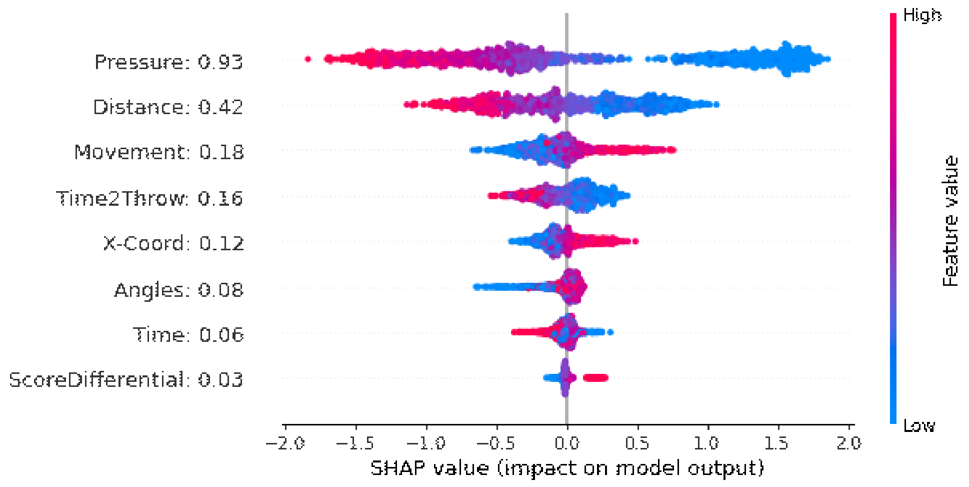

Figure 4.18 – Boosted Decision Tree Shapley Values for *first touch* success with tuned hyper parameters. Each predictions Shapley value individually plotted as a function of the feature value. Each observation can be deconstructed into their respective model impact using Shapley feature values.

the game) seem to provide the negative Shapley value. Showing why one does not at the average, but rather average the absolute Shapley contributions.

The color clusters for each of the individual features illustrate the inferences that have been mentioned throughout this analysis. For example, looking at the *Pressure* feature, we see that for both of the two success definitions that the positive Shapley values occur at low feature values (blue dots - low *Pressure* value) and the negative Shapley values occur at the high feature values (red dots - high *Pressure* value). Confirming our beliefs throughout this section which there is tangible proof that lower *Pressure* is a necessity to higher completion rates, regardless of how you define a Successful throw-in completion.

The *movement* feature exhibits interesting patterns for the *first touch* model, Figure 4.18. This will be discussed further in the next section or later work. The *movement* feature was made with the intent that more movement is made with the goal of creating less *Pressure* from the defenders near you and more space, thus creating an easier throw. This is shown to be correct as the positive Shapley value clusters occur with higher feature values (red dots - high values of movement) as opposed to the lower Shapley values clustering with lower feature values (blue dots - low values of movement).

Appendix A.2 delves further into the Shapley framework, specifically looking at two particular throw-ins, one which both definitions are deemed successful and one which both definitions are deemed unsuccessful. Individual predictions of the two throw-ins can be further inspected to see how each feature is individually impacting the model predictions made.

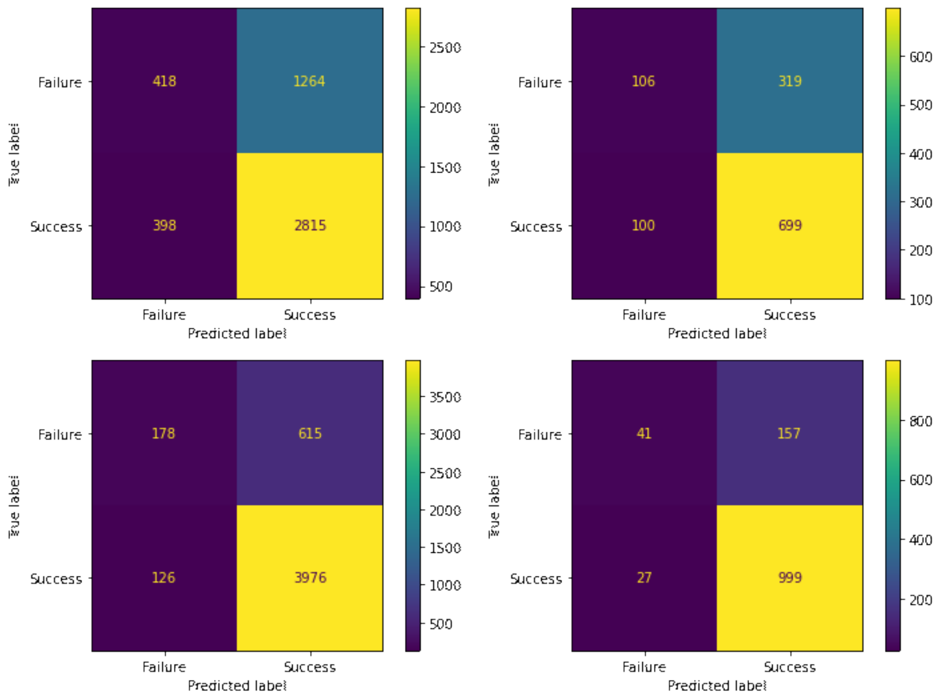

Figure 4.19 – Confusion matrix for the logistic regression model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing.

4.2.4 Extended Results

Next we will evaluate all of the models results through a confusion matrix and classification report (which provides precision/recall values), which were introduced in Section 3.2.1 (Figure 3.4). This framework is used again to inspect the precision and recall values, Equations 3.27 and 3.28 respectively. These numbers are important to look at because they evaluate how well the model is performing within the two classes (Successful and Unsuccessful throw-ins). Precision and recall values are evaluated for each of Logistic Regression, Decision Tree and the Gradient Boosted Decision Tree models.

Figure 4.19 illustrates the confusion matrices for the Logistic Regression model using both in-sample and out-of-sample testing for both of success definitions. Before analysing and drawing inference from the matrices presented it is important to look at the classification reports for the frameworks presented in Figures 4.19, 4.20 and 4.21. Table 4.4 illustrates the

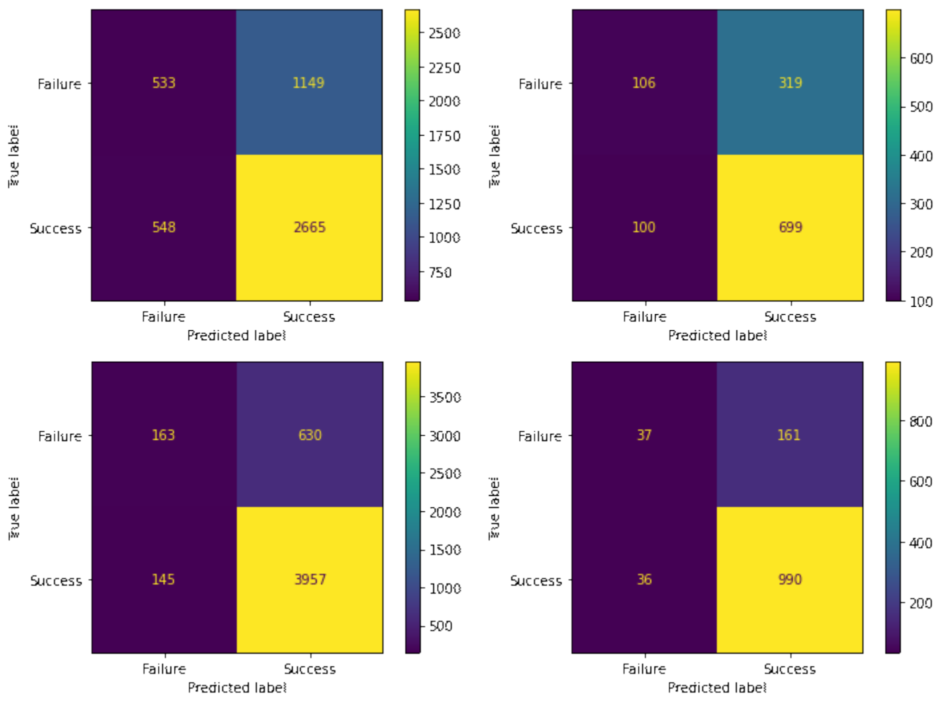

Figure 4.20 – Confusion matrix for the random forest model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing.

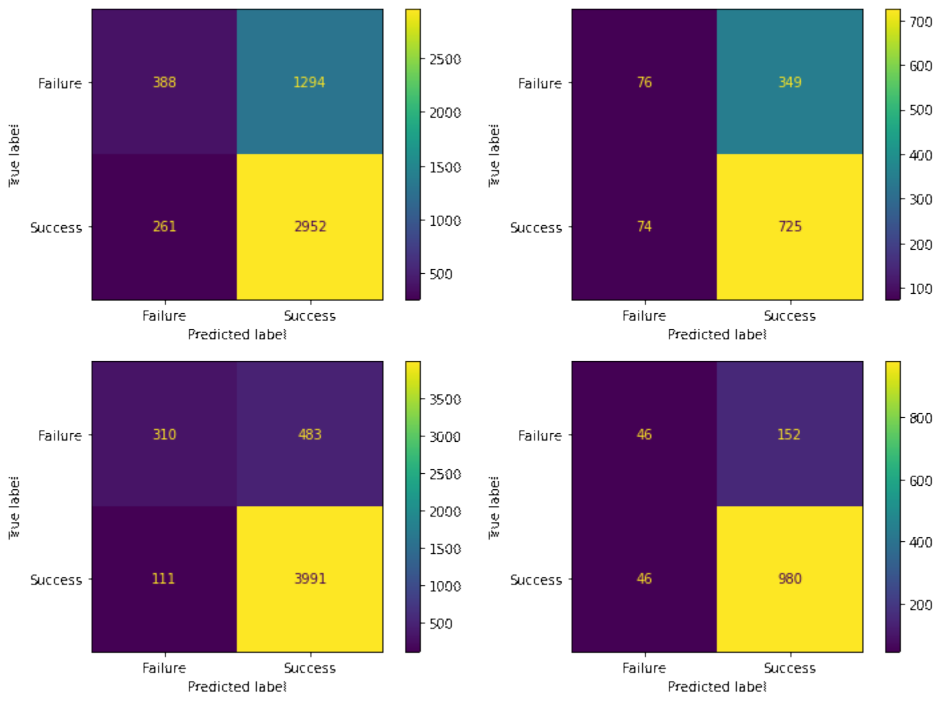

Figure 4.21 – Confusion matrix for the gradient-boosted decision tree model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing.

Model	Precision	Recall	F1-score
Logistic Regression	0.82/0.63	0.85/0.66	0.82/0.62
Random Forest	0.80/0.64	0.83/0.66	0.80/0.63
XGBoost	0.81/0.62	0.84/0.65	0.81/0.60

Table 4.4 – Classification report key metrics. Out-of-sample values are shown for each of the three models. First Touch in black and Threshold Retention in red.

classification report. Each of the tables portray all of the possible scenarios, models and definitions explored throughout the results section. The main metrics which the tables provides are the *precision*, *recall* and *f1-score*. The support column is the number of occurrences for that given column of that class (Successful/Unsuccessful throw-in).

A throw-in classified with an expected completion probability over 50% is classified as a *true positive* and a throw-in with an expected completion probability below 50% is defined as a *true negative*. A recall value (Equation 3.28) is effectively the ability to find all positive samples and a recall value of 1 is achieved if each throw-in is calculated to have an expected completion probability over 50%. The precision (Equation 3.27) is the ability of our model not to label a negative sample as positive. The F1-Score 3.29 generates the harmonic mean (weighted average) of the precision and recall scores. It is an important metric to look at, specifically for this case due to the uneven distribution of classes (Successful and Unsuccessful throw-ins).

The first observation when looking at Table 4.4 is the overall value difference for the two success definitions for each metric. There are a few factors as to why this is occurring, mainly the disproportionate number of occurrences in each of the two classes. For *threshold* success there is close to double the number of Successful throw-ins to Unsuccessful throw-ins in the training and test set. The *first touch* success has about five times the number of Successful throw-ins to Unsuccessful throw-ins in both the training and test sets. Although it does not explain everything, it does explain the large discrepancy in recall values.

Two methods will be explored to look at methods to improve this issue through threshold tuning (exploring various classification thresholds differing from 0.50). The intersection point is of interest as the generalized 50% relies on the assumption that the two classes are roughly the same size (and of equal importance). However, for data sets with class imbalance[4], the default threshold can result in poor performance. Thus one can choose a new threshold to try and minimize classification error.

The first method is to analyse the precision and recall values along different threshold values. This curve looks at the trade-off between precision and recall for different classification thresholds. High AUC scores represents both high recall and high precision but high precision relates to a low false positive rate and a high recall relates to a low false negative rate. Figures 4.22, 4.23 and 4.24 look at the precision-recall trade-off curves for the three models explored. Although the intersection point for each of the precision-recall graphs is different, they are rarely close to the 0.50 threshold which has been used. The methodology is not infallible by any means, however it does provide objective reasoning to a new threshold which one can analyse.

The second method to utilize threshold tuning is by optimizing along the AUC curve. Optimizing in terms of the threshold which provides the values of False Positive Rate (FPR) and True Positive Rate (TPR) that results in the upper-left corner of the curve. This point on the curve is the combination which minimizes the distance to the top left of the curve (which equates to a perfect AUC score of 1). This point satisfies threshold p, such that $TPR(p) = 1 - FPR(p)$. Therefore, the optimal threshold, is that which optimizes the following:

$$\hat{p} = \underset{p}{\operatorname{argmin}} |TPR(p) + FPR(p) - 1| \tag{4.2}$$

This is illustrated by plotting Equation 4.2 against various classification thresholds and verifying where the minimum occurs resulting in $\hat{p}$. Figures 4.25, 4.26 and 4.27 illustrates

[4]This data set does with a 8-1 ratio for *first touch* success and a 6-1 ratio for *threshold* success

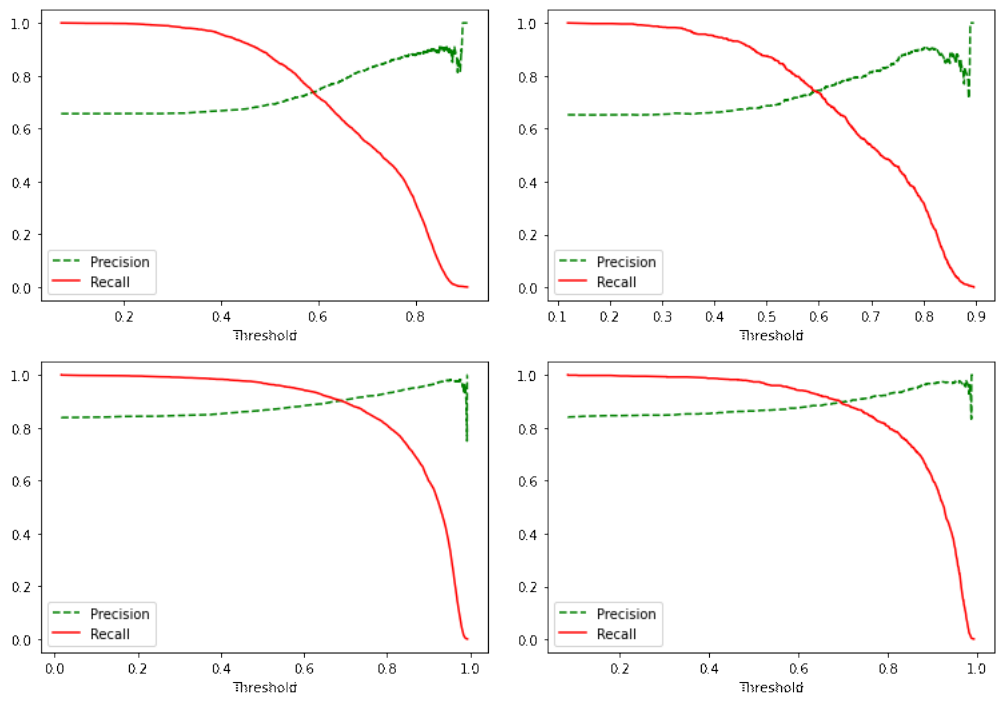

Figure 4.22 – Precision-recall trade-off graph for the logistic regression model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing.

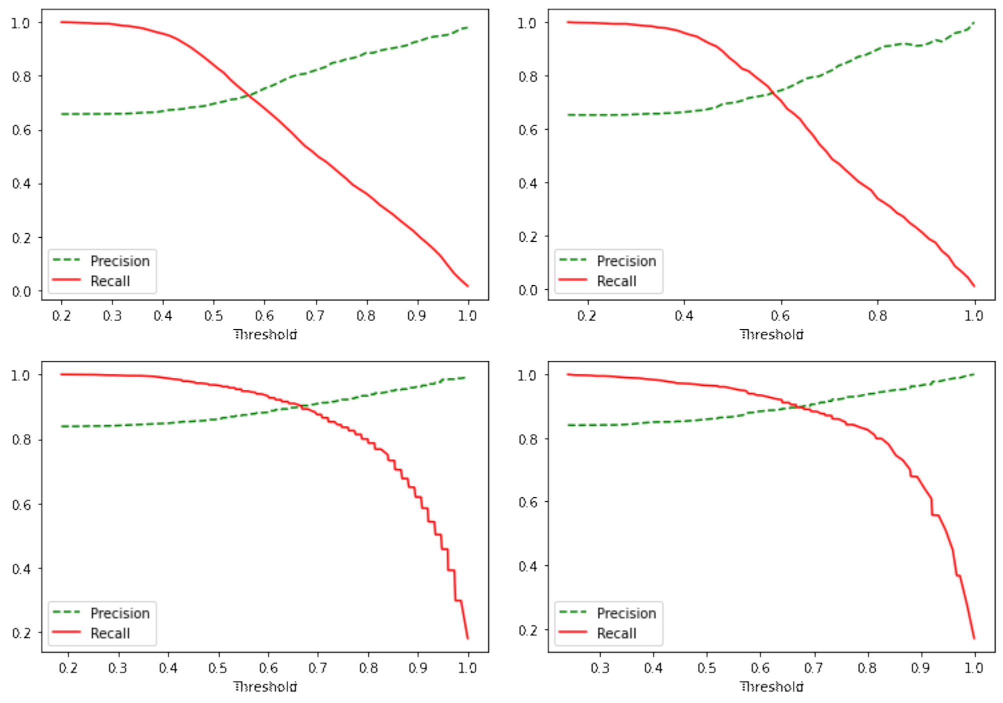

Figure 4.23 – Precision-recall trade-off graph for the random forest model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing.

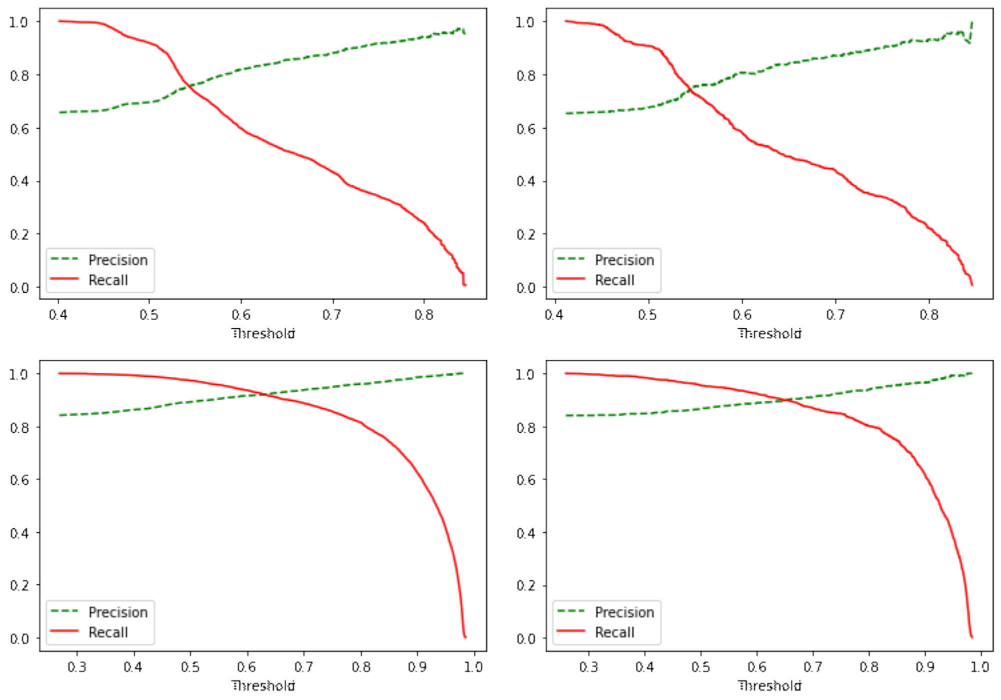

Figure 4.24 – Precision-recall trade-off graph for the gradient-boosted decision tree model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing.

the above equation for the three models explored.

As previously mentioned, using different thresholds from the traditional 50% is an interesting technique of creating new insights with logical reasoning for the predictions developed. The **optimal** thresholds found using this framework is once again not close to 0.50 for any of the scenarios, especially for the *first touch* success with values often close to 0.80. Using these new thresholds for the two methods presented, we can replace the 0.5 value which was previously used when analysing the precision and recall values in Table 4.4.

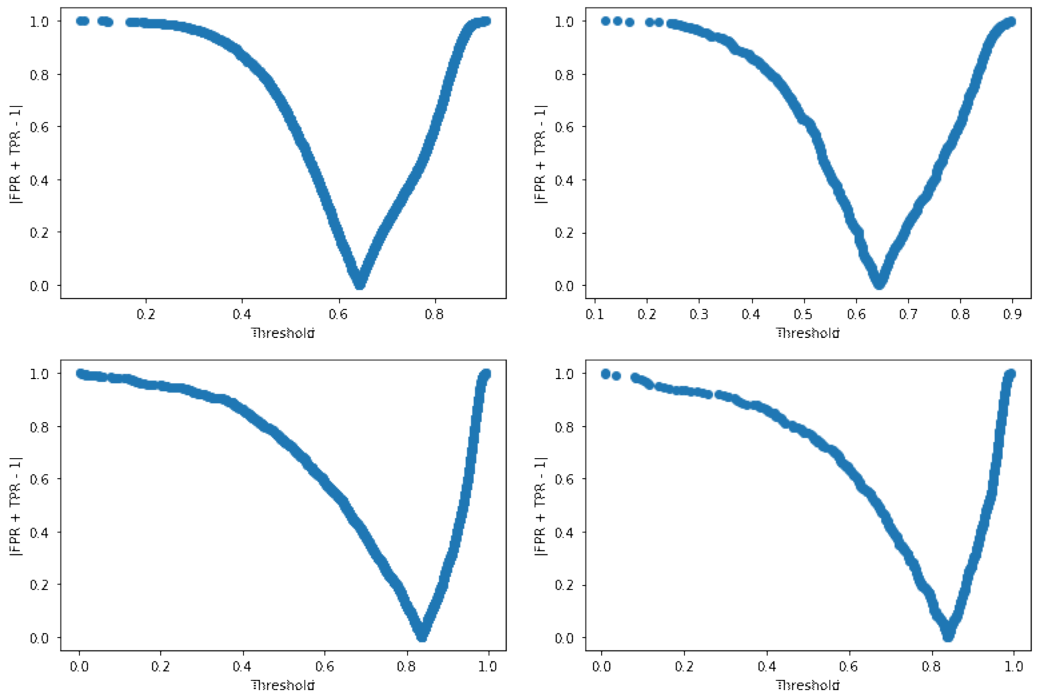

Figure 4.25 – Optimizing AUC graph for the logistic regression model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing. The optimal threshold following the order respectively from top-left to bottom right is 0.643, 0.643, 0.836, 0.835.

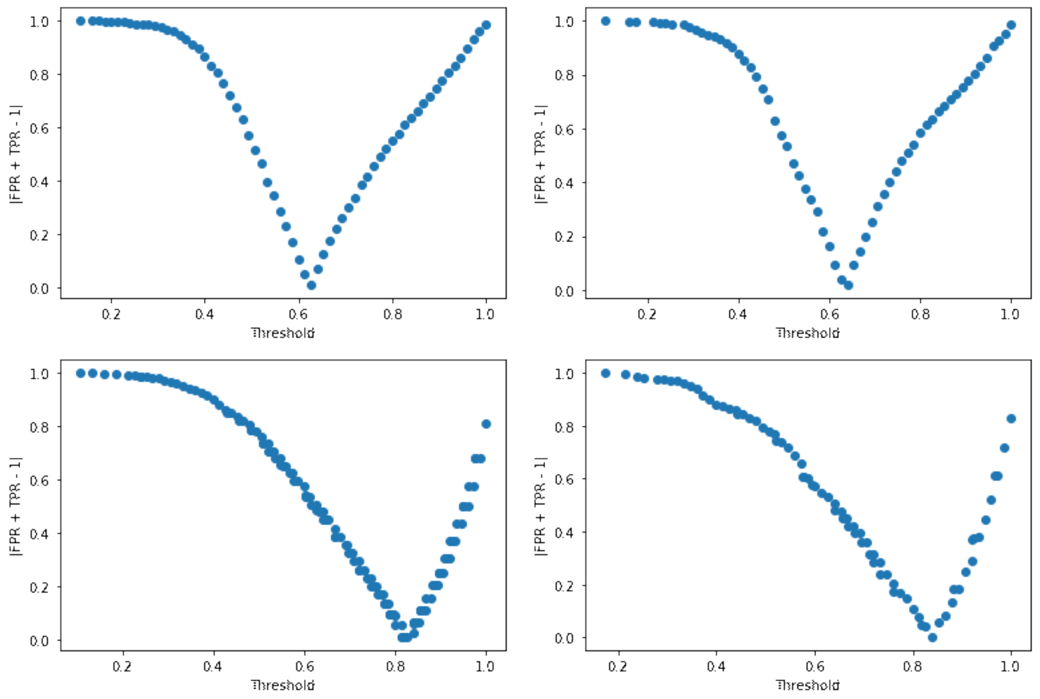

Figure 4.26 – Optimizing AUC graph for the random forest model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing. The optimal threshold following the order respectively from top-left to bottom right is 0.626, 0.64, 0.826, 0.84.

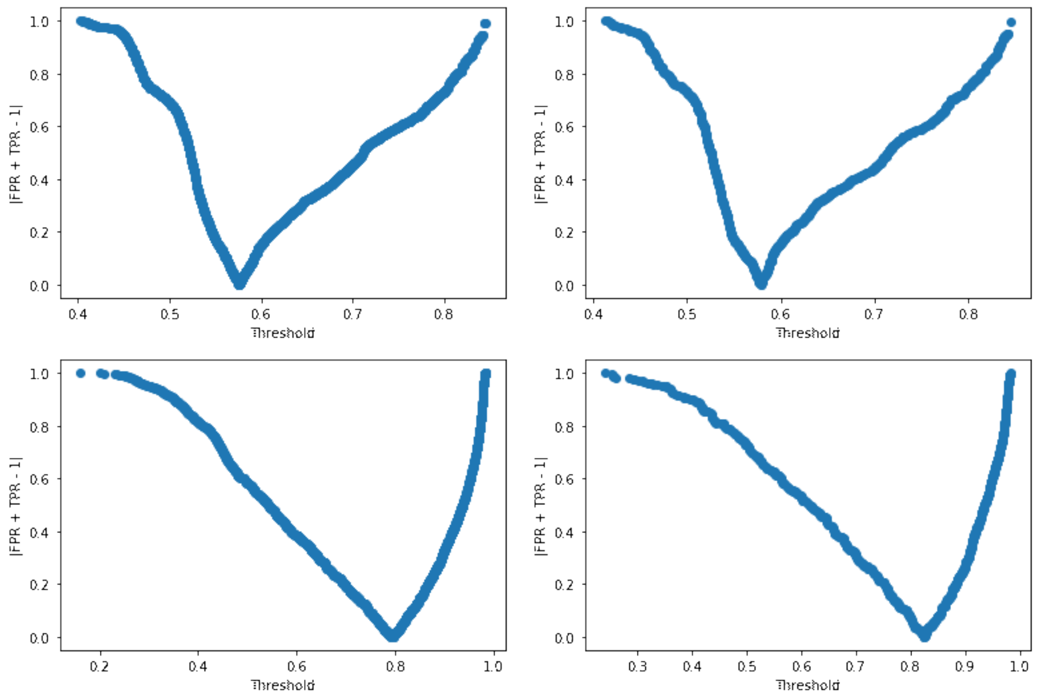

Figure 4.27 – Optimizing AUC graph for the gradient-boosted decision tree model. The top row is the *threshold* success and the bottom row is *first touch* success. The left-hand column is in-sample and the right-hand column is out-of-sample testing. The optimal threshold following the order respectively from top-left to bottom right is 0.576, 0.578, 0.794, 0.825.

Chapter 5

Practical Applications and Discussion

The goal of this thesis is to analyse and research a specific aspect of soccer, the throw-in, with the objective of searching for methods which can improve the completion rate of taking a throw-in. The analysis up to this point has been mathematically focused but yet it is important not to lose sight of the fact that this type of research can help a team improve on the field. After all, the sport of soccer is played on the field and not on spreadsheets. This section was created with the goal of taking the research above and manufacturing it into applicable resources for teams to take to the field.

5.1 Segmentation Analysis over Clustered Throw-Ins

The two graphics below are important to analyse, as they provide a visual representation of how the two definitions differ throughout the pitch. The idea was taken from Stone et al. [8] whose paper was focused on the impact that throw-ins have on individuals and team success as a whole. Before going in-depth into throw-in research, one can plot the success rates for various lengths, directions and areas of the pitch which Stone et al. [8] also plotted in his paper - but we will use the DFB data throughout this analysis. Following the ideology of Stone, the meanings of each type of throw are defined below in Table 5.1. For this exercise, the definitions of *Successful* combines both *SuccessfullyComplete* and *Successful* from the definitions mentioned in Section 2.2.4.

Using the definitions explained in Table 5.1, the success rates are then explored in the following figures. The figures below are taken directly from Stone et al. [8] which draws throw-in data from the 2018–2019 Premier League season as well as the results from this analysis using the data given by the DFB to draw inference and comparisons.

Overall, at a first glance when comparing the two numbers one can see that the success rates are quite similar for the two definitions at each location. It is important to note that the attacking direction for Stone is right to left as opposed to this thesis being left to right, resulting in the locations of all the success rates being flipped. Comparing the two first contact success rate figures (Figure 5.1 and 5.3) one can see that the general pattern of success rates reducing from short to low throws occur along with the lowest success rates occurring on long throws in the offensive direction.

Similar trends are followed for the possession retention success definition in Figures 5.2 and 5.4. The lowest success rates occur at the forward throws for both the attacking and defensive zone. The results match Stones for all areas of the field. One of the more interesting developments which was found by both Stones research and this research paper is that the attacking half's success rate is actually higher than the defending half for the long throws.

5.2 Impact of Time-to-Throw-In on Success Rates

Although the feature values have been examined above in Chapter 4, it was done at a macro level for correlation plot and inspected deeper in relation to model results. This section

Category	**Operational Definition**
First Contact	*Successful*: A player from the same team which throws the ball into play makes first contact with the ball post throw-in without an opposition player making contact. *Unsuccessful*: A player from the opposition team which throws the ball into play makes first contact with the ball post throw-in. *Success percentage*: Calculated by dividing the number of successful first contacts in a category (i.e. short) by the total number of actions (Successful + Unsuccessful) performed in that category and multiplying by 100.
Possession retention	*Successful*: The ball is retained in possession (as defined in Section 2.2.4) for 7 seconds from the point in which the ball is thrown. *Unsuccessful*: The ball possession is lost (as defined in Section 2.2.4) with in 7 seconds from the point in which the ball is thrown. *Success percentage*: Calculated by dividing the number of successful possessions retained in a category (i.e. short) by the total number of actions (excluding those this did not achieve a successful first contact) performed in that category and multiplying by 100.
Throw-in length	*Short*: The ball was thrown a distance between 0–10 yards (0–9.1 m). *Medium*: The ball was thrown a distance between 10–20 yards (9.1–18.2 m). *Long*: The ball was thrown a distance of 20 yards or longer ($>$ 18.2 m).
Throw-in direction	*Forward*: The ball is thrown between 0–60° in reference to the sideline towards the offensive goal. *Lateral*: The ball is thrown between 60°–120° in reference to the sideline. *Backward*: The ball is thrown between 120°–180° in reference to the sideline towards the defensive goal.

Table 5.1 – Operational definitions for throw-in lengths, directions and outcome variables. Based on the definitions from Stone et al. [8].

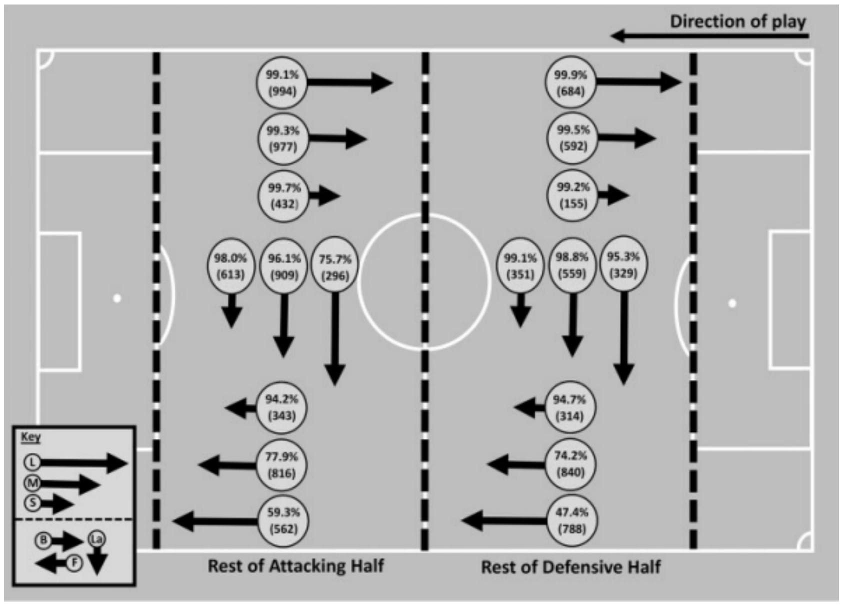

Figure 5.1 – First contact success rate (percentage) based on pitch location, throw-in direction and throw-in length taken directly from Stone et al. [8]. Number of occurrences shown below success rates.

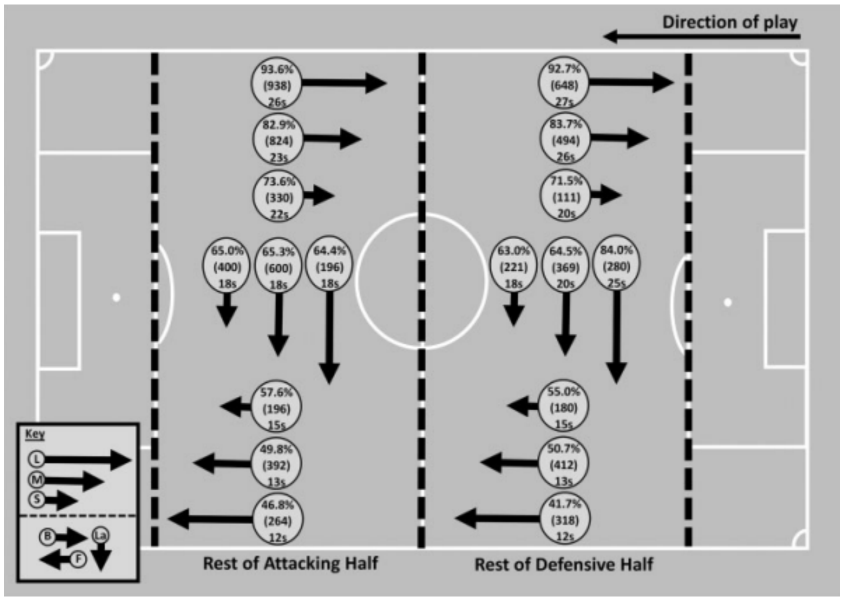

Figure 5.2 – Possession retained success based on pitch location, throw-in direction and throw-in length. Percentage success, absolute values and mean time in possession. Taken directly from Stone et al. [8]. Number of occurrences shown below success rates.

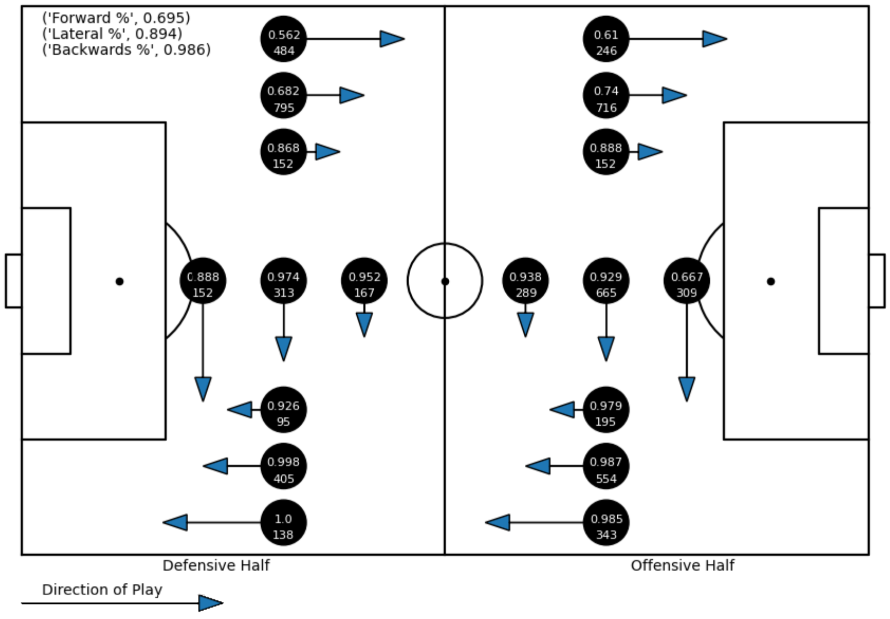

Figure 5.3 – First contact success rate based on pitch location, throw-in direction and throw-in length using data for this paper. Number of occurrences shown below success rates.

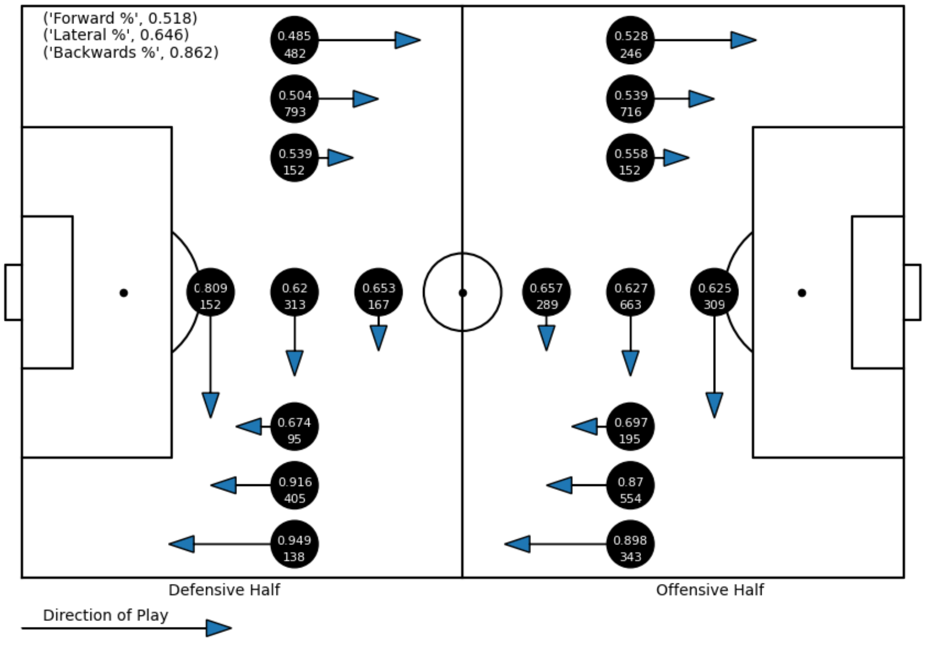

Figure 5.4 – Possession retained success rate based on pitch location, throw-in direction and throw-in length using data for this paper. Number of occurrences shown below success rates.

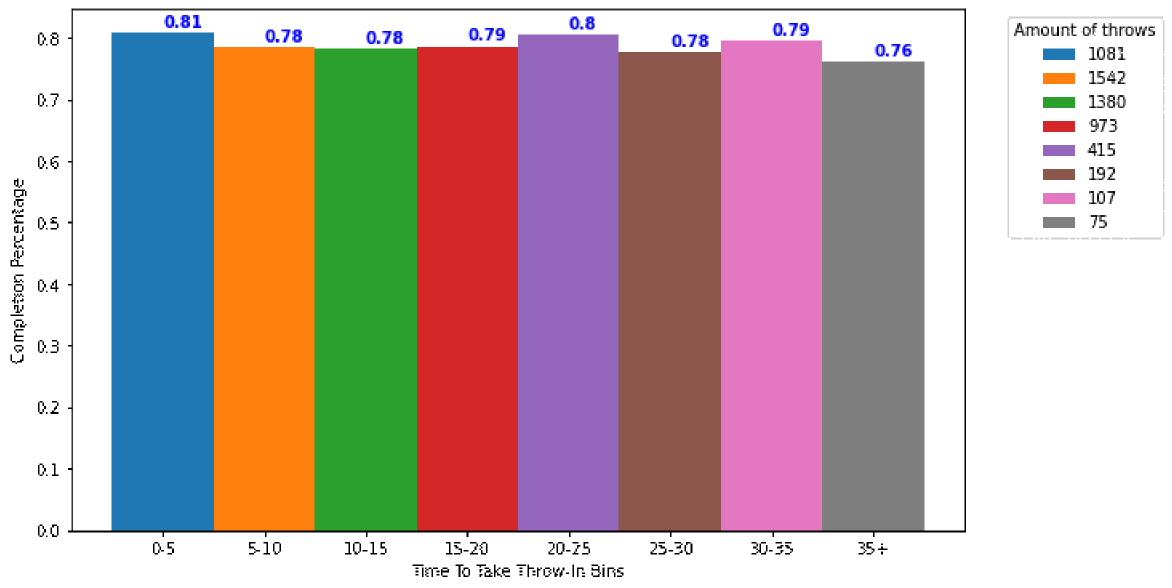

Figure 5.5 – Binning throw-ins every 5 seconds by their time to take a throw-in, using *first touch* success definition. The number of throw-ins for each bin is shown on the right.

will further dig into the feature results as it relates to the two success definitions and how inferences can be made from it. In previous sections it was hypothesized that certain feature values would lead to higher or lower success rates and this section will aim to debunk those assumptions.

The first assumption which will be inspected is regarding the feature *Time to throw-in* and whether faster throw-ins are wanted and if there is a certain value which is best. The two figures below (Figures 5.5 and 5.6) look at the throw-in completion rate for the two success definitions, categorized by every 5 seconds after the throw-in is thrown. Each of the two figures follow a clear descending completion percentage by bin, except for Figure 5.6 which has a sudden jump in completion percentage at 30-35 seconds. Although the decrease is not monumental, it does point to throw-ins being completed quicker being optimal for the *threshold*. This can be due to how they are being defined, regardless of a threshold, one can "easily" throw to a receiver and hit him thus becoming a *first touch* success. However, throwing to a receiver which leads to a successful *threshold* completion, 7 seconds later, requires much more variables which *first touch* does not.

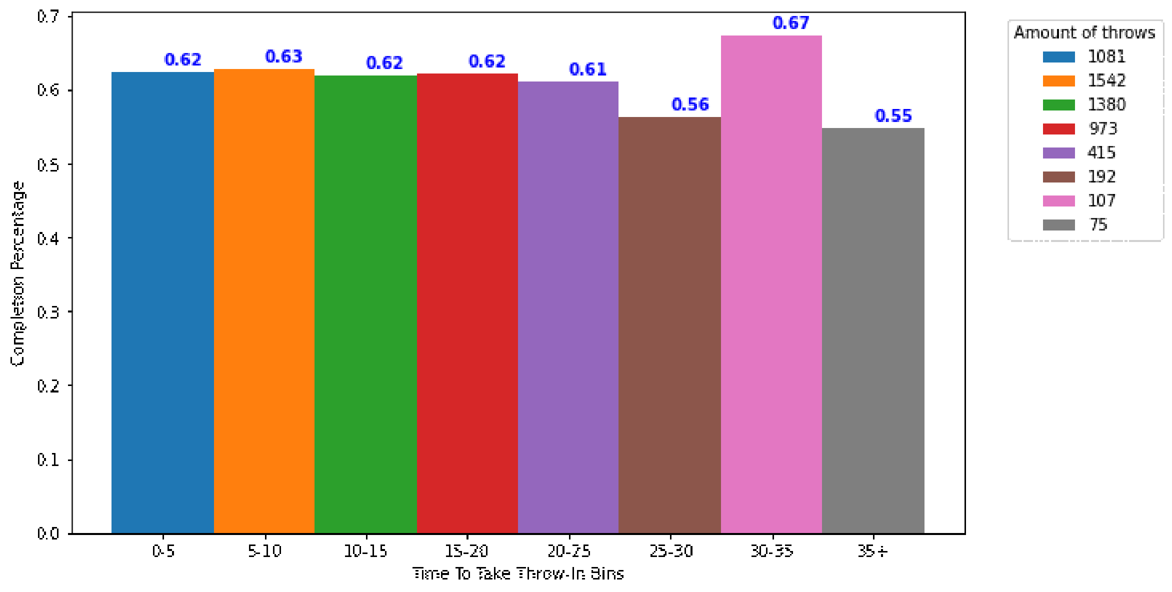

Figure 5.6 – Binning throw-ins every 5 seconds by their time to take a throw-in, using *threshold* success definition. The number of throw-ins for each bin is shown on the right.

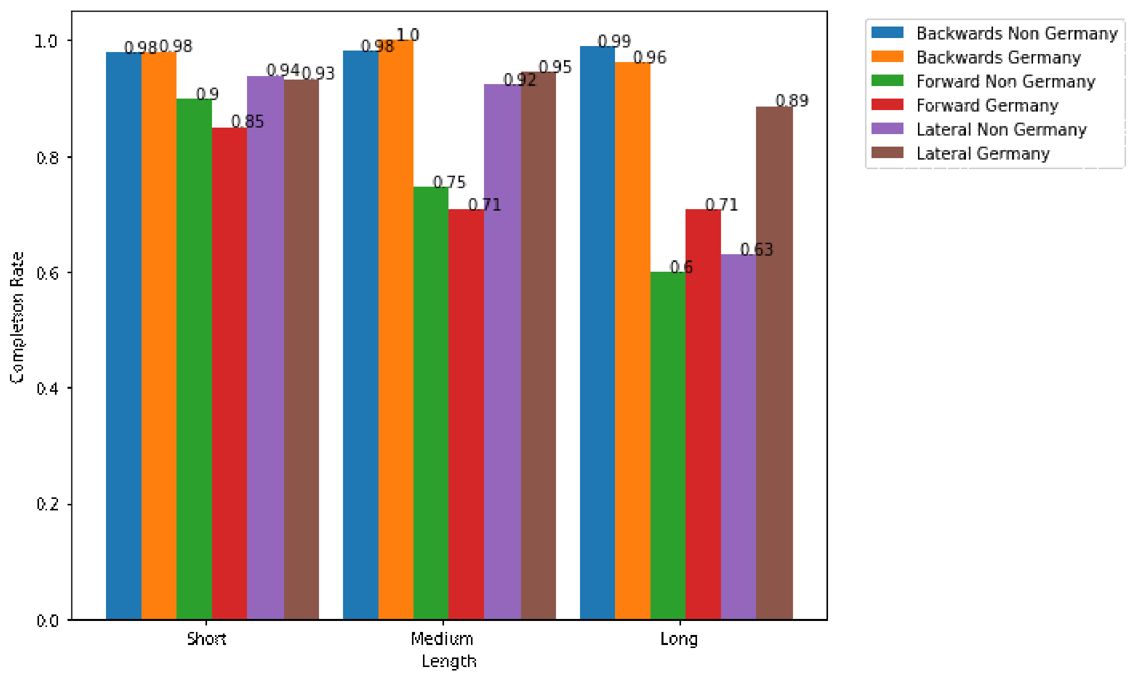

Figure 5.7 – Evaluating **Attacking** throw-ins (towards the opposing teams net). Split data set to compare the DFB (Germany) and the competitor's throw-in completion percentage at each length and direction (definitions from Table 5.1). Evaluated using *first touch* success.

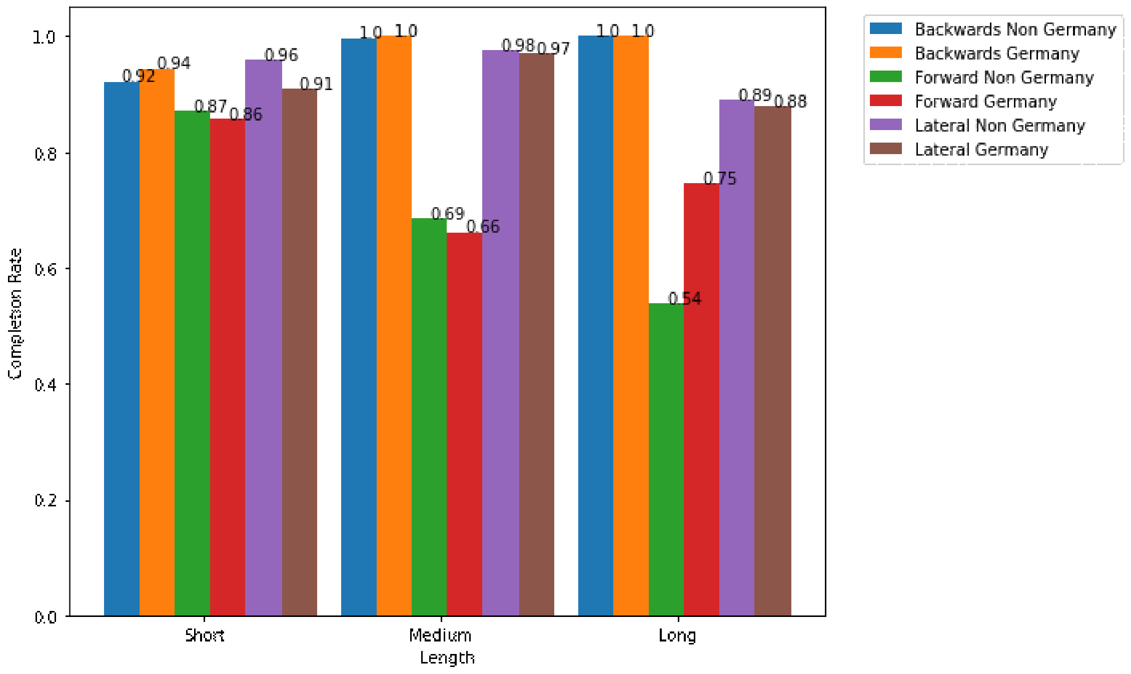

Figure 5.8 – Evaluating **Defensive** throw-ins (towards the throwing teams net). Split data set to compare the DFB (Germany) and the competitor's throw-in completion percentage at each length and direction (definitions from Table 5.1). Evaluated using *first touch* success.

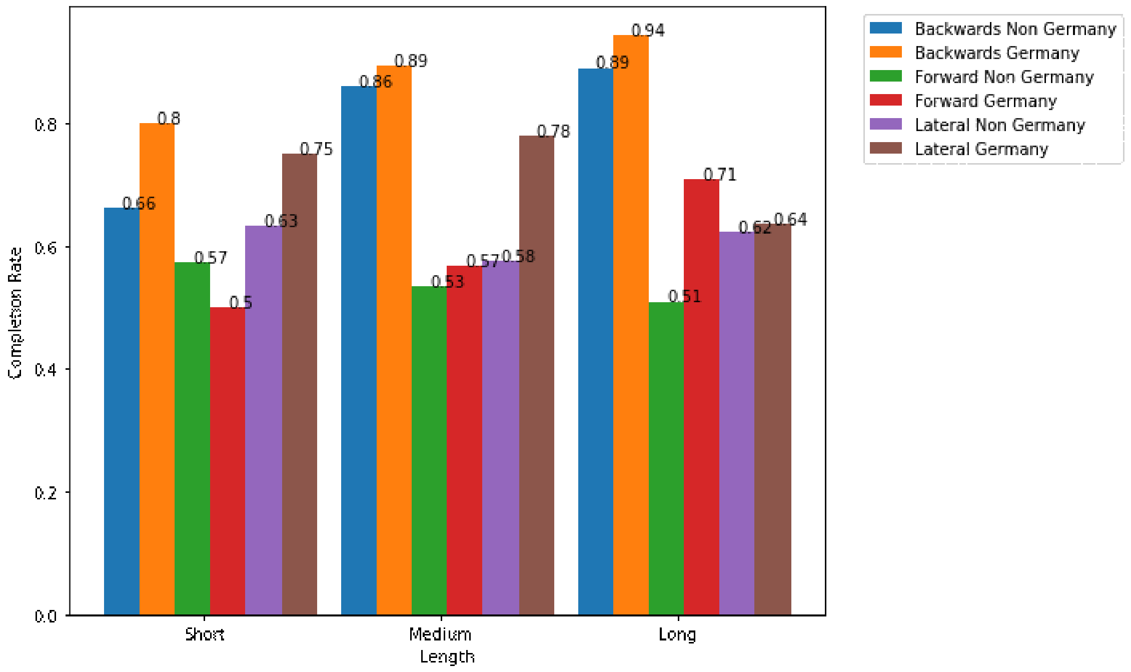

Figure 5.9 – Evaluating **Attacking** throw-ins (towards the opposing teams net). Split data set to compare the DFB (Germany) and the competitor's throw-in completion percentage at each length and direction (definitions from Table 5.1). Evaluated using *threshold* success.

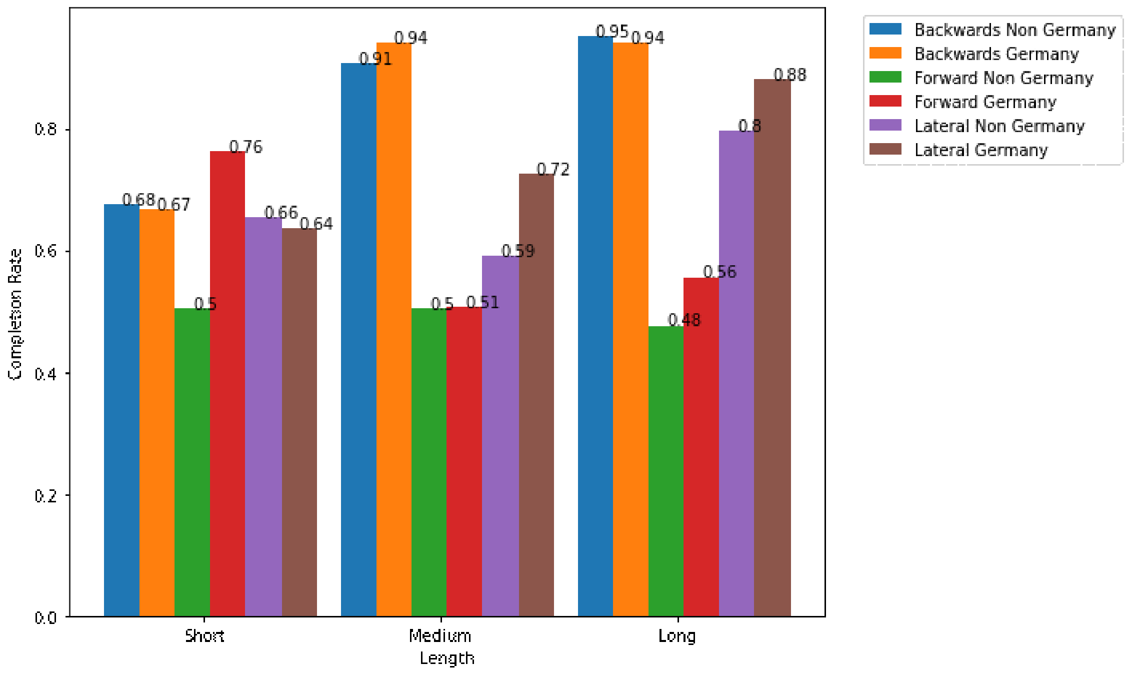

Figure 5.10 – Evaluating **Defensive** throw-ins (towards the throwing teams net). Split data set to compare the DFB (Germany) and the competitors throw-in completion percentage at each length and direction (definitions from Table 5.1). Evaluated using *threshold* success.

5.3 Comparison of the DFB and the Competition

This section is written with the intent of evaluating the performance of the DFB versus the competition along with providing insight into throw-in analysis to further assist the team. To summarize how the DFB is performing for throw-ins in each section of the field, the following tables are presented. Table 5.2 illustrates the areas of over performance by the DFB and Table 5.3 illustrates the areas of under performance. These are taken from the threshold retention success values (Figure 5.4) but the same trends hold for first touch success values.

It is clear the DFB is outperforming the competition for many throw-in types (for both success definitions as well). Further work can investigate evaluating more throw-ins and further separating by age, competition and gender to get a better idea of talent evaluation.

Location	**DFB/Non-DFB**	**Sample Size**
Long/Forward/Offensive	0.71/0.51	24/222
Long/Forward/Defensive	0.56/0.48	54/428
Short/Forward/Defensive	0.75/0.50	21/131
Short/Backwards/Defensive	0.80/0.66	18/77
Medium/Lateral/Offensive	0.78/0.58	168/495
Medium/Lateral/Defensive	0.72/0.59	69/244

Table 5.2 – **Over performing throw-ins.**

Location	**DFB/Non-DFB**	**Sample Size**
Short/Forward/Offensive	0.50/0.57	40/157

Table 5.3 – **Under performing throw-ins.**

5.3.1 xThrow-in in Practice

Below are examples of the model in practice. Looking at all the feature values for each of the potential receivers and using the model to evaluate the predicted completion percentage, according to the two response definitions.

These two throws were not chosen randomly, the first throw was deemed successful according to both first touch and threshold retention and similarly the second throw was deemed unsuccessful according to the two definitions. This provides a visual component which teams can use to evaluate scenarios throughout a game to see the probabilities they are facing for the given throw-in. The general takeaway is the predicted probabilities are higher for first touch as opposed to threshold retention (which is of course expected).

The primary reason for the selection of these two throw-ins is to illustrate the impact of pressure on the expected probability for the receiver in each of the respective throws. The pressure value for Figure 5.11 is 1.23 and the pressure value for Figure 5.12 is 1.22 (in effect the same). The predicted probabilities using first touch success is 0.903 and 0.548 versus the predicted probabilities using threshold success being 0.509 and 0.478. This (along

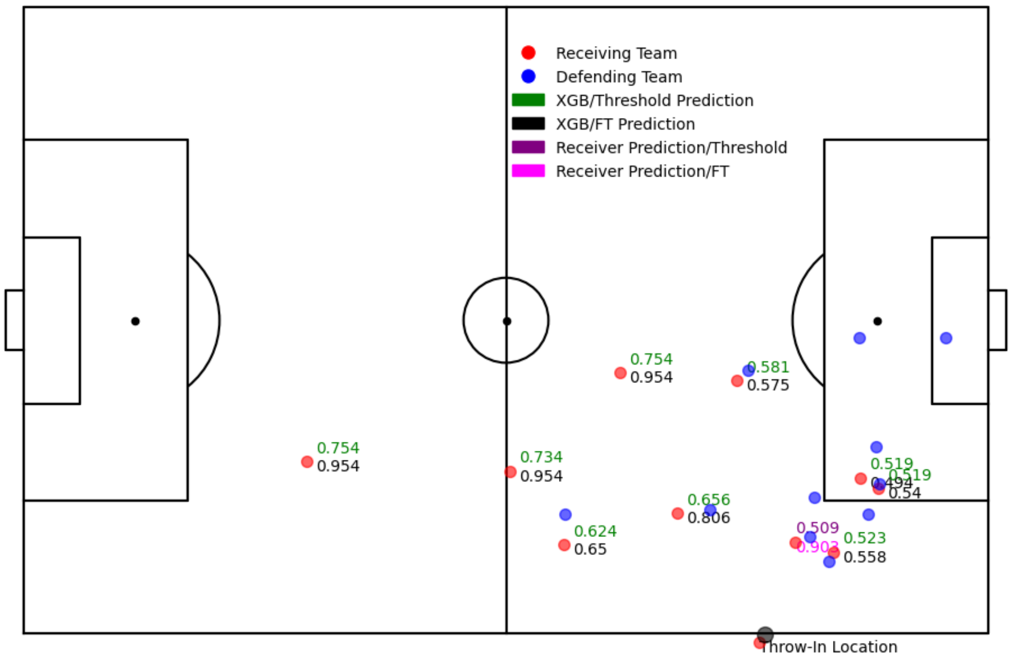

Figure 5.11 – Example of Throw-in Predictions created for all the potential receivers using the XGBoost model. Both Definitions were deemed **Successful**.

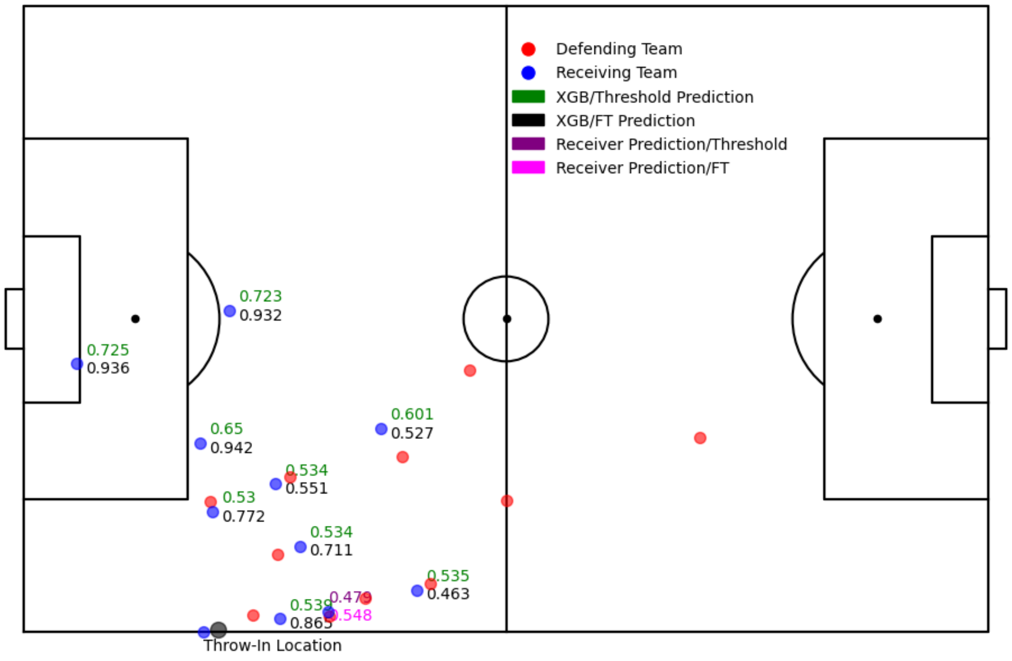

Figure 5.12 – Example of Throw-in Predictions created for all the potential receivers using the XGBoost model. Both Definitions were deemed **Unsuccessful**.

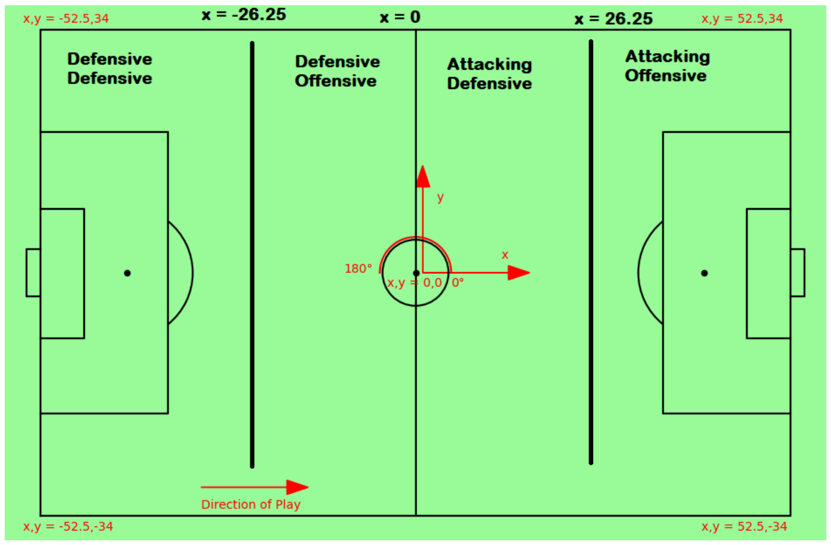

Figure 5.13 – Segmented Field Regions (cut into 4 equal regions along the length of the field).

with material shown below) illustrates the main objective of this section which is, given the pressure value is high, one is flipping a coin on whether possession retention is maintained (regardless of the other feature values).

5.3.2 Segmentation Analysis Exploration

To form an opinion if feature values present more importance for different areas of the field, the field was segmented. The field was divided into four sections along the length of the field. Figure 5.13 illustrates the four sections along with the names which will be referenced further on.

Using the XGBoost model (Gradient Boosting Decision Tree) with the same hyper parameters as above (Table 3.1) the feature importance was evaluated for all four sections (using both success definitions). Figure 5.14 illustrates the segmented results using first touch success and Figure 5.15 illustrates the segmented results using threshold retention success.

The first observation is the overwhelming importance of the pressure feature no matter the

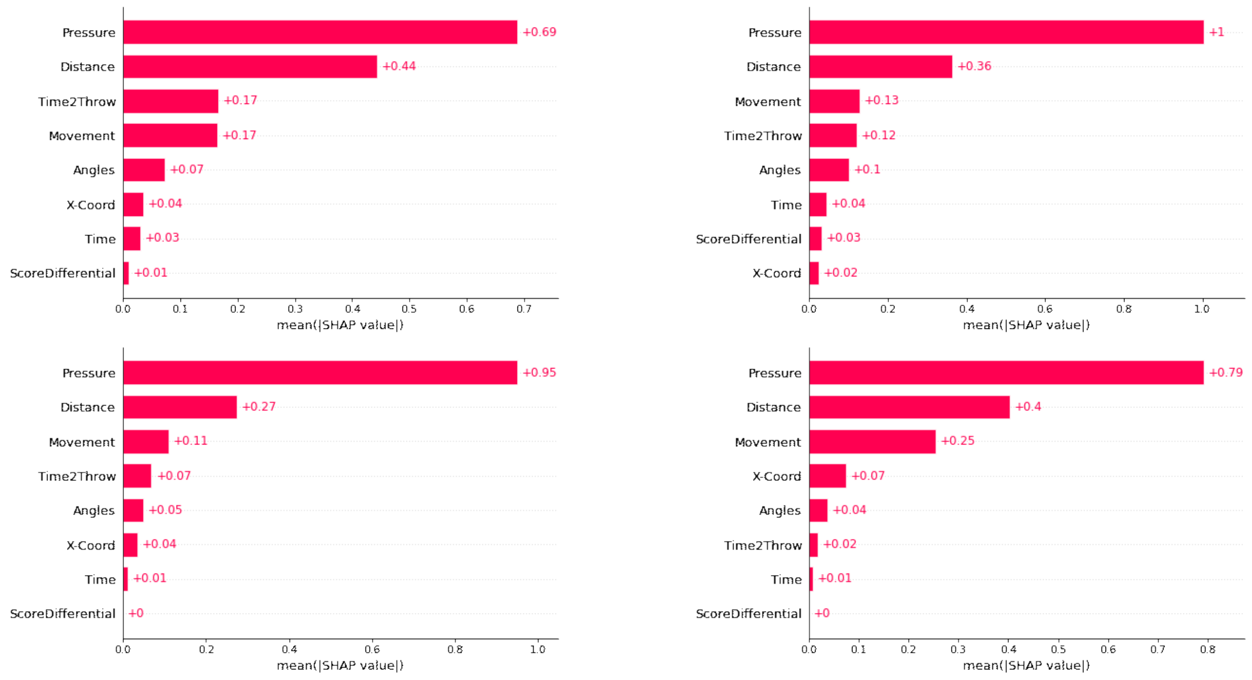

Figure 5.14 – Using **First Touch Success** the top left is **DD**, top right is **D0**, bottom left is **AD** and bottom right is **AO**.

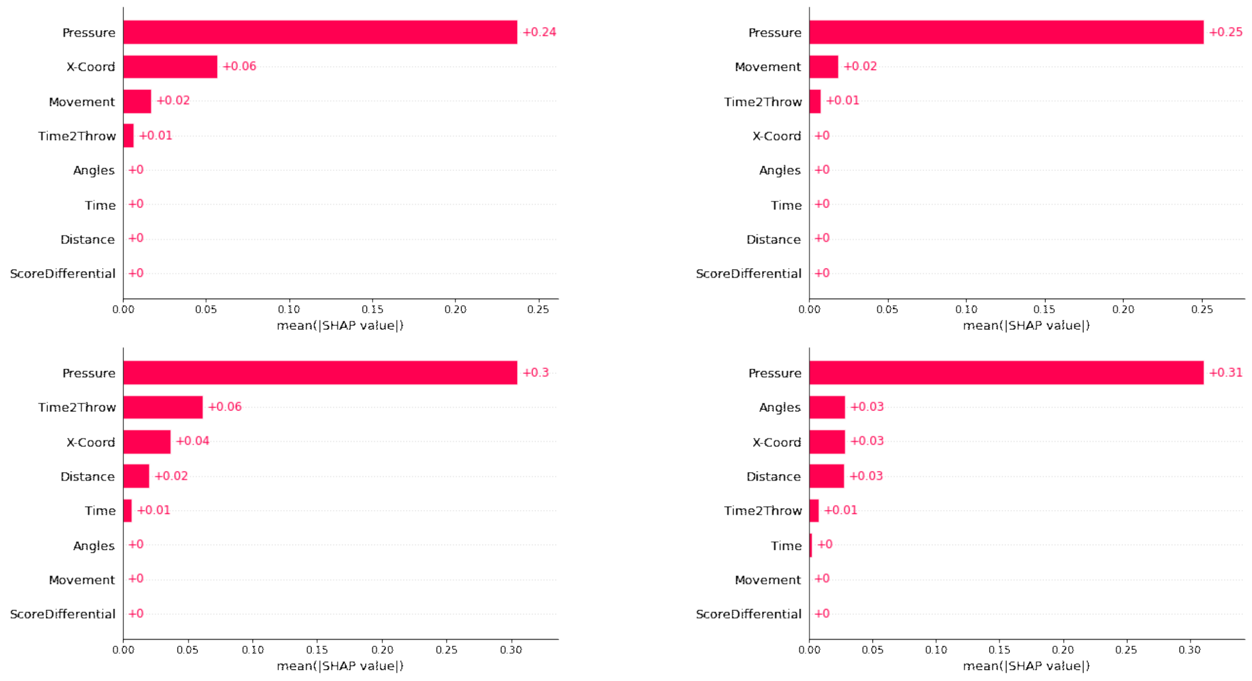

Figure 5.15 – Using **Threshold Retention Success** the top left is **DD**, top right is **D0**, bottom left is **AD** and bottom right is **AO**.

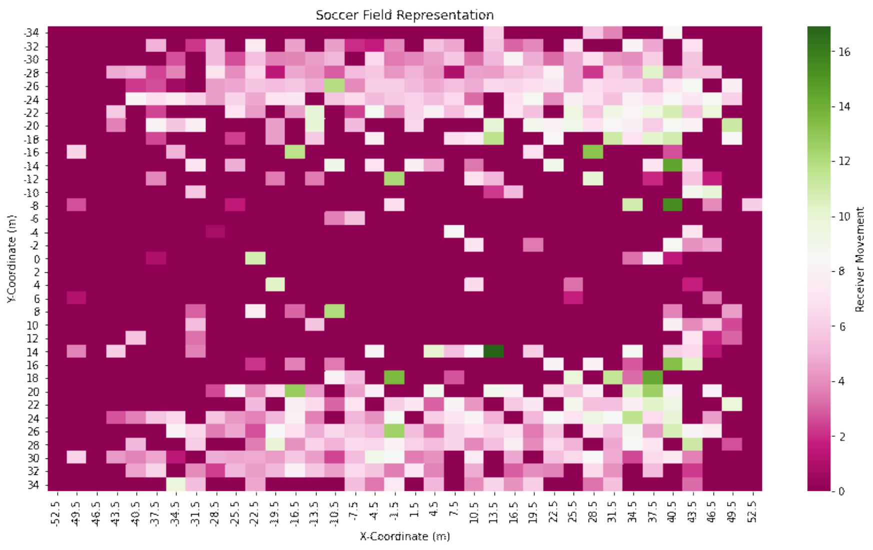

Figure 5.16 – Heat Map of Receiver Movement. Soccer field cut up into 2 by 3 (meter) quadrants where the average **Receiver Movement** experienced in that quadrant is illustrated.

success definition or area of the field, it remains the most important. This discrepancy is most pronounced for the possession retention success where it is effectively all that matters. This leads to the ideology that unless scoring or taking on unnecessary risk is truly needed for the throw, it is optimal to take the easier throw-in (low pressure).

The second important observation from the figures is the importance of receiver movement, specifically at the Attacking Offensive (AO) zone for first touch success (bottom right plot of Figure 5.14). As previously mentioned, the importance values are so small for threshold retention (Figure 5.15) it is hard deduce inferences. One can see with a feature importance value of 0.25 for receiver movement in the Attacking Offensive (AO) zone, it nearly doubles the receiver movement feature value in the other three areas. This is expected as you are closest to the opposing net and therefore it is the area of most importance for creating opportunities to score. Creating receiver movement will lead to better opportunities, as throw-ins into the opposing box generally lead to a shot, header, or passes possibly resulting to a dangerous opportunity.

To evaluate how much receiver movement players are creating within the data set, Figure

Success Definition	**Top-1 Rate**	**Top-3 Rate**	**Top-5 Rate**
First Touch	0.213	0.299	0.367
Possession Retention	0.133	0.170	0.271

Table 5.4 – Model accuracy of defining **safest** receiver. Percent of time the highest (along with within Top-3 and Top-5) predicted completion percentage is actually thrown to according to the XGBoost model.

5.16 was developed. The figure is a heat map for receiver movement, with green color illustrating a higher receiver movement value as opposed to a pink color illustrating a low receiver movement value. According to the Shapley feature importance figure for the attacking offensive zone (furthest right region of Figure 5.16) in order to optimize completion rates, receiver movement is of upmost importance. This is not occurring however on this figure, as if it would, there would be far more green quadrants closer to the opposing net. This is a clear area which teams can improve upon to optimize throw-in completion (and following dangerous opportunities on goal). Appendix A.3 further discusses Figure 5.16. The plot is cut up into successful throw-ins and unsuccessful throw-ins (using the first touch definition). Due to the data imbalance one cannot make definite conclusions, but most of the receiver movement is occurring for the successful throw-ins leading to the hypothesis of greater receiver movement leading to success.

To get an idea of how often the receiver which the model predicts to be the **safest** receiver is actually thrown to, Table 5.4 was developed. Safest is defined as the highest predicted completion percentage according to the XGBoost model. Both success definitions are analysed along with how often the ball is thrown to a receiver in the top 3 and top 5 highest predicted completion percentages.

The results from Table 5.4 illustrates one of the themes throughout this analysis, that it is easier to predict first touch success. A clear trend is shown of being approximately 10% more accurate at identifying the receiver using first touch as opposed to possession retention. To

Success Definition	**Top-1 SD**	**Top-3 SD**	**Top-5 SD**
First Touch	0.078/-0.004	0.093/-0.021	0.096/-0.035
Possession Retention	0.038/0.001	0.1141/-0.007	0.1247/-0.03

Table 5.5 – In accordance with Table 5.4, the average *Score Difference* for each of these bins is illustrated. Black means the average Score Difference is within range and Red means the average Score Difference is not within range. A positive number indicates the throwing team is leading and a negative number means the team is trailing.

form an idea of assessing risk within these throw-ins, Table 5.5 was developed.

Assessing risk, from the perspective that the model (in its current state) produces an expected completion percentage of a throw-in, but it does not provide an answer as to which teammate one should direct the throw-in to create a high-percentage goal opportunity (risk/reward payoff). The black number is the average score difference given that the safe receiver was able to be identified in that range (from Table 5.4) and the red number is the average score difference given that the safe receiver was not able to be identified in that range.

The results provide an interesting and expected story. For both success definitions, when the identified receiver (or top 3 and top 5) is in accordance to being the one thrown to (black number), it is with an average positive score difference. On the contrary, when the model is not able to identify the safe receiver (red number) it is with a average negative score difference. This leans into the idea that when in a position of power (i.e. leading the match) and no risk needs to be taken, the model predicts and identifies (with relatively good accuracy) the receiver as no risk is needed. Alternatively, when trailing the match, and therefore taking the easier throw-in will not net the team any possible future gains (dangerous opportunity on net) the model is not able to identify to whom the ball is thrown to. The discrepancy from in versus out of range (black vs red color) is most pronounced for possession retention, which is expected given the feature importance of pressure at all areas of the field (Figure 5.15). Due to low pressure on leading throws (black color) the throw is easily completed and possession kept (signifying a successful possesion retention).

Chapter 6

Conclusion

The objective of this thesis was to delve into the research of the throw-in aspect of a soccer match to provide possible improvements soccer teams can make in the future. Throw-in models were developed to produce an expected probability for any given throw-in, with the idea that teams can evaluate various scenarios within a soccer match to evaluate given historical precedence the likelihood a throw-in is completed (according to two definitions of completion). Binary classification models were powered through novel tracking data (explained in Section 2.2.2) with two methods for creating the definition of a successful throw-in: *first touch* success and *possession retention* success (explained in Section 2.2.4). Three classification models were used to evaluate 6119 throw-ins of which 83.8% were a successful *first touch* success (5128 instances) and 65.6% were a successful *threshold retention* success (4012 instances). The binary classifiers used were Logistic Regression, Random Forest and Boosted Decision Tree.

In Chapter 4, all the results for each of the models were discussed with the general consensus after examining metrics such as AUC score, Precision, Recall and log-likelihood values that predictions are more accurate for a first touch success as opposed to a threshold retention response. This does not mean that a threshold retention response variable is not valuable to future research, it simply means given the current feature values it performed worse. Throughout the 7 seconds from the point of the ball being thrown to the threshold being met, much unforeseen (or predicted) randomness occurs. This response variable is of more importance to teams as maintaining possession means that one has the opportunity to score, and more importantly, the other team does not. Future work can explore ways to further utilize and analyse possession retention by improved features and unforeseen models.

Using a first touch response variable did create satisfactory predictions with noteworthy accuracy, giving the model a strong framework at predicting throw-in completion percentage. Figure 5.11 and 5.12 provide illustrations of real-life applications which teams and coaching personnel can use to evaluate throw-ins.

Throughout the various studies and analyses, it is clear that given the current definition of a throw-in success, pressure is practically all that matters in completion. Whether it be direct low pressure values or utilizing receiver movement to create low pressure. Furthermore, maintaining possession can be completely dictated by pressure. If the pressure value for a given receiver is high, you are flipping a coin on predicting if possession retention is kept.

The method proposed in this thesis aims at creating a generalized model for throw-ins, in the footsteps of xPass (expected passes) and xG (expected goals) models presented by Anzer and Bauer [11]. This goal was met with encouraging success at predicting throw-in success, the next step will be incorporating a risk/reward function. This will hopefully answer the question: *Who* should I throw to (given the game context)? Not **where** can I throw it to optimize completion.

Bibliography

1. *Sports Market Report* https://www.globenewswire.com/fr/news-release/2021/03/18/2195540/28124/en/Global-Sports-Market-Report-2021-to-2030-COVID-19-Impact-and-Recovery.html. Accessed: 2022-03-27.

2. Eaves, J. S. A history of sports notational analysis: a journey into the nineteenth century. *International Journal of Performance Analysis in Sport* **15,** 1160–1176 (Dec. 2015).

3. Haygarth, A. *Lillywhite's cricket scores and biographies of celebrated cricketers, from 1746 to [1898]* (Longman, London, 1925).

4. Rottenberg, S. The Baseball Players' Labor Market. *Journal of Political Economy* **64,** 242–258 (June 1956).

5. Lindsey, G. R. Tatistical Data Useful for the Operation of a Baseball Team. *Operations Research* **7,** 197–207 (Apr. 1959).

6. Szymanski, S. Sport Analytics: Science or Alchemy? *Kinesiology Review* **9,** 57–63 (Feb. 2020).

7. *Society for American baseball researchKernel Description* https://sabr.org/sabermetrics. Accessed: 2022-03-27.

8. Stone, J. A., Smith, A. & Barry, A. The undervalued set piece: Analysis of soccer throw-ins during the English Premier League 2018–2019 season. *International Journal of Sports Science & Coaching* **16,** 830–839 (Feb. 2021).

9. Hammoudeh, M. A. A., Alsaykhan, M., Alsalameh, R. & Althwaibi, N. Computer Vision: A Review of Detecting Objects in Videos – Challenges and Techniques. *International Journal of Online and Biomedical Engineering (iJOE)* **18,** 15–27 (Jan. 2022).

10. Manafifard, M., Ebadi, H. & Moghaddam, H. A. A survey on player tracking in soccer videos. *Computer Vision and Image Understanding* **159,** 19–46 (June 2017).

11. Anzer, G. & Bauer, P. A Goal Scoring Probability Model for Shots Based on Synchronized Positional and Event Data in Football (Soccer). *Frontiers in Sports and Active Living* **3.** doi:10.3389/fspor.2021.624475. https://doi.org/10.3389/fspor.2021.624475 (Mar. 2021).

12. *A New World of Performance Insight from Broadcast Tracking TechnologyKernel Description* https://medium.com/skillcorner/a-new-world-of-performance-insight-from-video-tracking-technology-f0d7c0deb767. Accessed: 2022-03-27.

13. Stein, M. *et al.* Visual Soccer Analytics: Understanding the Characteristics of Collective Team Movement Based on Feature-Driven Analysis and Abstraction. *International Journal of Geo-Information* **4,** 2159–2184 (Oct. 2015).

14. *Capturing and visualizing large scale human actionKernel Description* https://www.csc.kth.se/~sullivan/actvis/Docs/proj_proposal.pdf. Accessed: 2022-03-27.

15. Andrienko, G. *et al.* Visual Analysis of Pressure in Football. **31.** ISSN: 1384-5810. doi:10.1007/s10618-017-0513-2. https://doi.org/10.1007/s10618-017-0513-2 (2017).

16. Hastie, T., Witten, D., Tibshirani, R. & James, G. *An introduction to statistical learning: With applications in R* (Springer, 2017).

17. Hastie, T., Friedman, J. & Tisbshirani, R. *The elements of Statistical Learning: Data Mining, Inference, and prediction* (Springer, 2017).

18. Hanley, J. A. & McNeil, B. J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* **143,** 29–36 (Apr. 1982).

19. *Understanding Confusion Matrix* https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. Accessed: 2022-05-31.

20. Strumbelj, E. & Kononenko, I. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems* **41,** 647–665 (2013).

Appendix A

A.1 Event Data

	Column	Description
General Information (available before the throw-in is thrown)	Competition	Competition the throw-ins were collected in
	Match Day	Day the match was executed
	Match ID	ID of the match
	Game Title	Match title
	X-Coordinate, Y-Coordinate	x- and y-coordinate of where the throw-in leaves the thrower's hands
	Minute	Minute when the throw is played
	Match Clock	Minute and second of the throw
	Frame	Frame number respective to the time of the throw
	Reception Frame	Frame number respective to the time of reception of the throw
	Player ID1	ID of the player taking the throw
	Player ID2	ID of the receiver of the throw (whether it be the intended receiver or not)

	Intended Receiver ID	ID of the intended receiver of the throw
	Player1 Team ID	ID of the team taking the throw
	Player2 Team ID	ID of the team of the player who receives the ball
	Intended receiver team ID	ID of the team of the player which the throw was intended for
	Club 1	Club abbreviation for the team taking the throw, I.e. three letter code
	Club 2	Club abbreviation for the team of which the player receives the throw, I.e. three letter code
	Player 1 shirt number	Shirt number of the thrower of the ball
	Player 2 shirt number	Shirt number of the receiver of the ball
	Score difference	Score difference at the time point of the throw-in between the team taking the throw and the defending team

	Column	**Description**
Information on the throw outcome (from the offensive perspective)	Evaluation	Indicates success of a ball played: ●*SuccessfullyComplete*: If team of player receiving the ball and that of player playing ball are identical ●*Unsuccessful*: If player receiving the ball is on opposing team of player receiving the ball ●*Successful*: If the team playing the ball stays in possesion despite no player having received the ball (e.g., if a player of the team

		playing the ball is fouled before he can successfully receive the ball).
	Going Right	True if throw is played on right pitch side
	Subtype	•*Cross*: long throw into the penalty box •*Pass*: short, played throw
	X Reception Coordinate	x-coordinate where the throw-in was received by the discovered true receiver
	Y Reception Coordinate	y-coordinate where the throw-in was received by the discovered true receiver
	X Intended Receiver Coordinate	x-coordinate of the location where the intended receiver was standing at the time of throw
	Y Intended Receiver Coordinate	y-coordinate of the location where the intended receiver was standing at the time of throw
	Throw time	Seconds after ball goes out of bounds too thrown
	Pressure on Receiver	Indicates the amount of pressure exerted on the receiver of the ball
	Summed Movement	Using our tracking data, we see how players (in meters) from frame to frame. The distance is calculated as the summed movement from the throw-in frame to the frame of reception.
	Angle to ball	The angle measured on a 0 to 180 degree scale 180 degree scale (normalized). Angle taken from the thrower to all potential receivers.
	Distance of throw	Euclidean distance of how far the throw is from the location

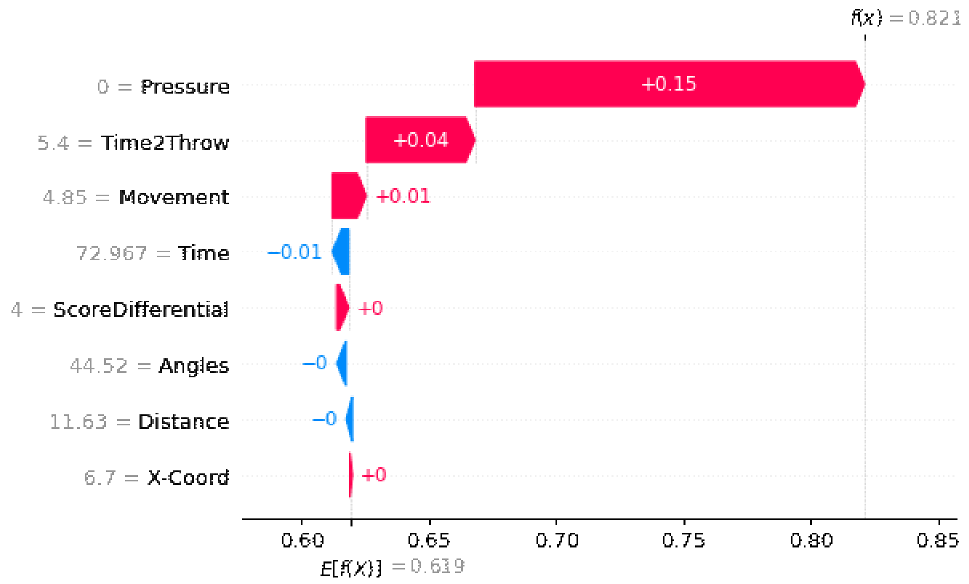

Figure A.1 – Construction of Shapley value for a single prediction using *threshold* success - which was deemed Successful according to *threshold* success.

A.2 Shapley Individual Throw-In Predictions

To further analyse these Shapley framework, an individual prediction can be further inspected to see how each feature is individually impacting the prediction our model makes. First, a Successful throw-in (for both definitions) is inspected below in Figures A.1 and A.2. The figure illustrates all of the feature values for that individual throw-in along the vertical axis, $E[f(X)]$ is the mean prediction from the out-of-sample test set using the respective success definitions and $f(x)$ on the top right of the figures is the prediction made for this individual throw. This plot is directly illustrating Equation 3.31, $\phi_i(\hat{f})$ is the contribution of the $i - th$ feature on the individual prediction (for example the value $+0.11$ for *Pressure* in Figure A.1), $\hat{f}(x)$ (0.98 for Figure A.1) is the prediction and $E[\hat{f}(x)]$ (0.836 for Figure A.1) is the mean prediction for the respective test sample (in this case looking at *first touch* success). The predicted value on the top right according to the Shapley framework (0.98 for Figure A.1) is created by taking the expected prediction value (0.836 for Figure A.1) and adding all of the feature contributions (red/blue numbers).

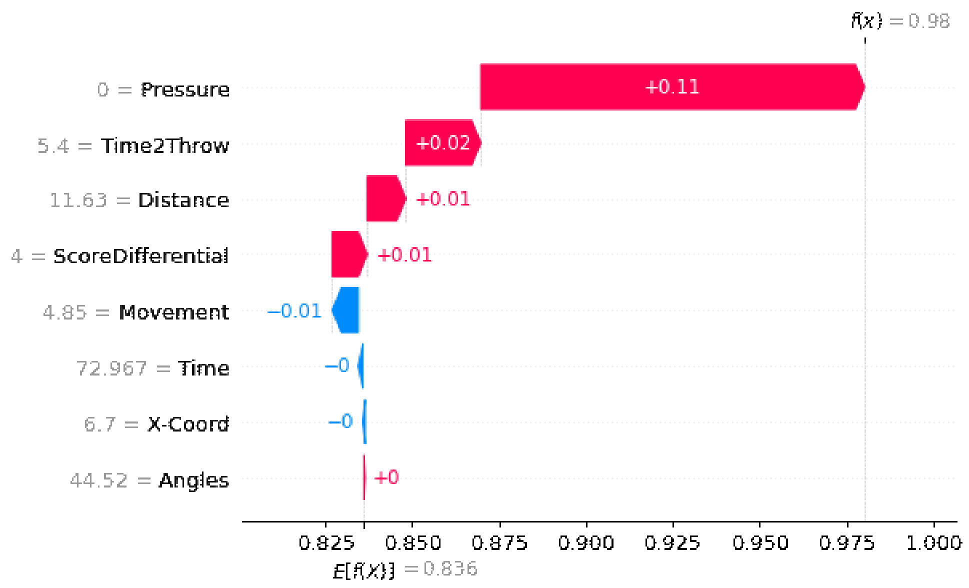

Figure A.2 – Construction of Shapley value for a single prediction using *first touch* success - which was deemed Successful according to *first touch* success.

This throw-in was chosen to inspect for a few reasons. First, it was a success for both definitions, and second, the *Pressure* value was 0 for both of them. With a *Pressure* value of 0, one can see how much value it provides to the prediction compared to all of the other feature values for both of the success definitions. Given the current set of features, if the *Pressure* value is 0, one can speculate with great certainty that the throw was most likely completed - unless some very unlikely scenario occurred. It is of note to see that the *time to throw* feature provides the second most value, but it is so small and relatively random that it is hard to draw any major conclusions from this.

Next, an example in which both of the two success definitions are deemed Unsuccessful will be examined using this same Shapley framework. Figures A.3 and A.4 portray the same Unsuccessful throw for *threshold* and *first touch* success definitions respectively using the previously mentioned Shapley Equation 3.31. One can see that the throw was incomplete (Unsuccessful) but the predicted probability (for both definitions) are still reasonable and not dramatically close to 0 with values of 0.527 and 0.458 for *threshold* and *first touch* success definitions respectively. The conundrum with the *Distance* feature is presented here and it is

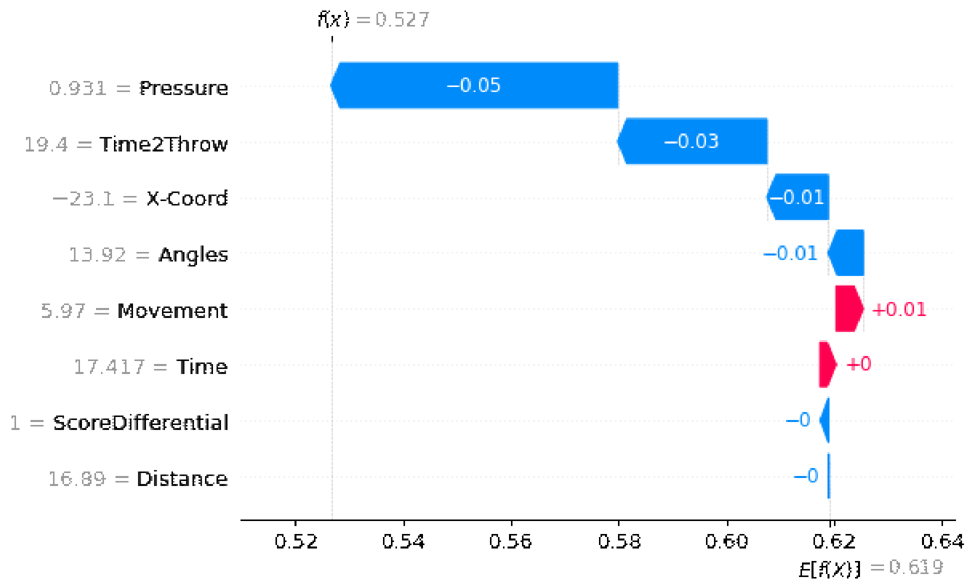

Figure A.3 – Construction of Shapley value for a single prediction using *threshold* success - which was deemed Unsuccessful according to *threshold* success.

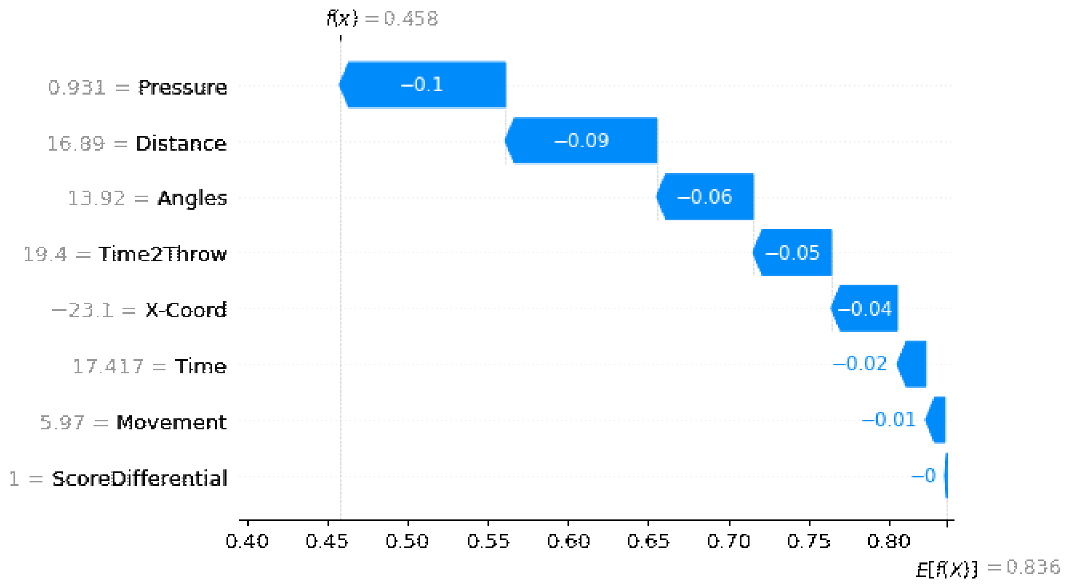

Figure A.4 – Construction of Shapley value for a single prediction using *first touch* success - which was deemed Unsuccessful according to *first touch* success.

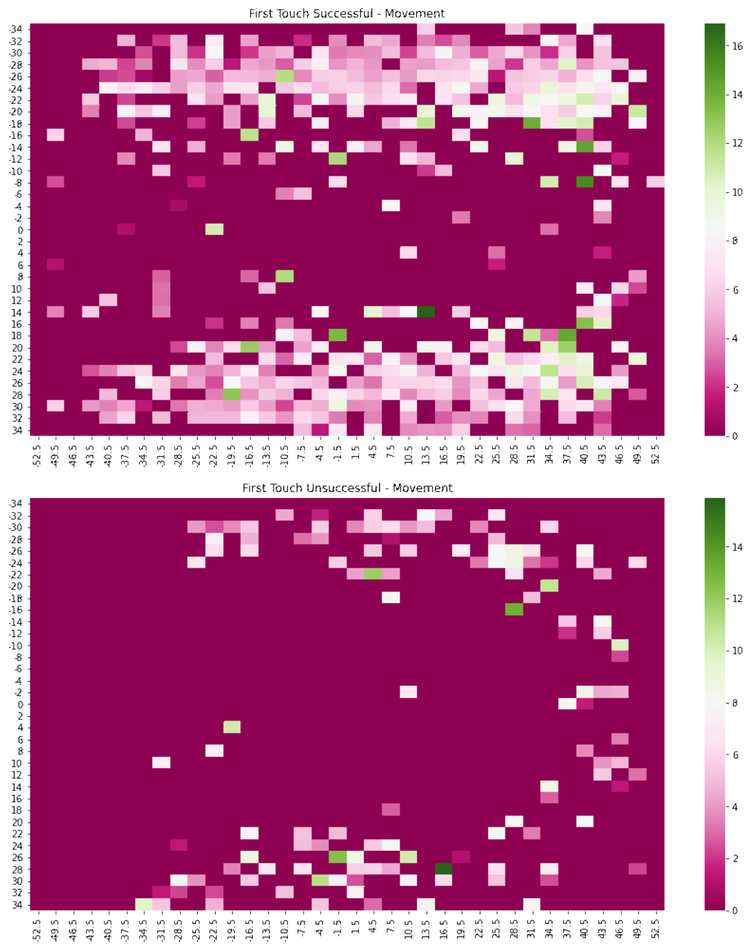

Figure A.5 – Heat Map of Receiver Movement split in **Successful** Throw-Ins (Top) and **Unsuccessful** Throw-Ins (Bottom).

interesting to inspect. Looking at Figure A.4 we see that *Distance* provides the second most impact (albeit negative) versus Figure A.3 where it provides no impact, a stark difference. Besides those two features, it is hard to draw many conclusions beyond the varying impact *Distance* has from the two definitions and how dramatic the impact *Pressure* provides to the model - whether it be positively or negatively.

A.3 Plots

To further investigate receiver movement, the heat map methodology was used once again by developing one strictly for successful throw-ins and unsuccessful throw-ins (using the first touch definition). This further illustrates the point that the vast majority of receiver movement is occurring for the successful throw-ins (top of Figure A.5) as opposed to lower (or no) receiver movement occurring for unsuccessful throw-ins (bottom of Figure A.5). This leads to the natural progression that the more movement created, which is completed to evade defenders (and lower pressure), the more likely the throw-in is to be completed. The idea behind receiver movement is related to pressure, it used as a tactic to evade the defender to an area not crowded which creates low pressure and thus higher throw-in completion rates.

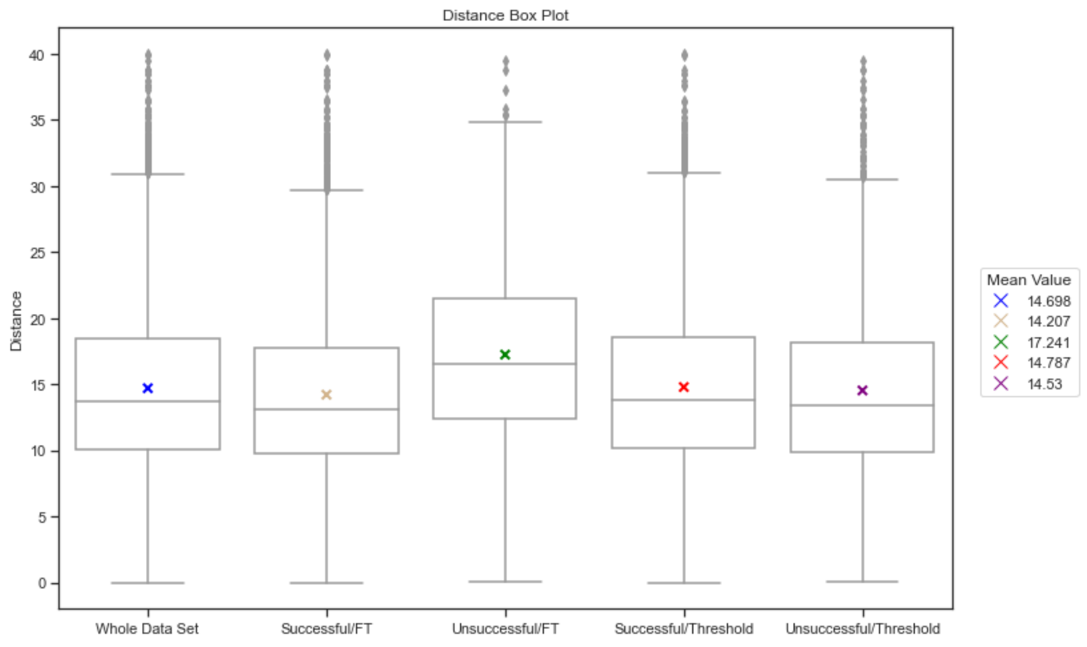

Figure A.6 – Distance Box Plot

A.4 Scatter Plot of Feature Combinations

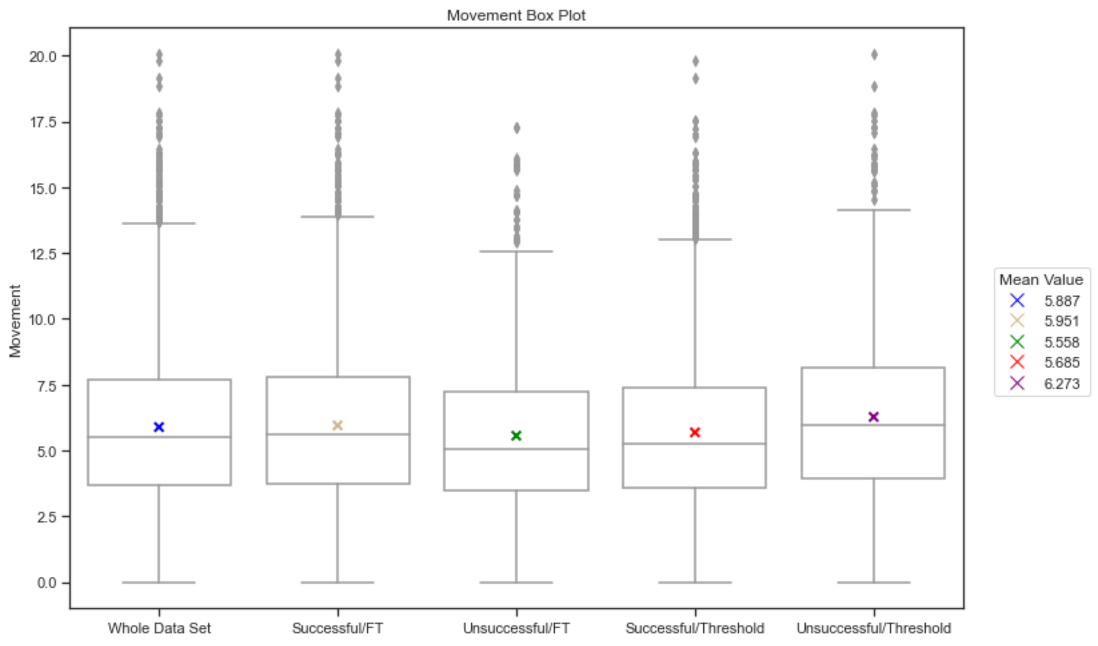

Figure A.7 – Movement Box Plot

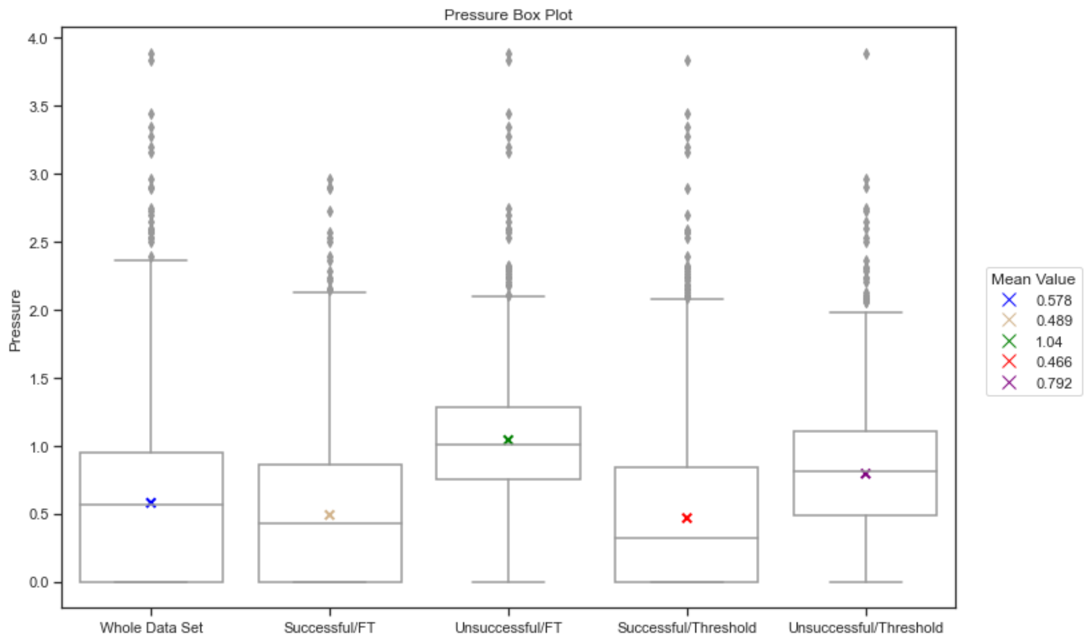

Figure A.8 – Pressure Box Plot

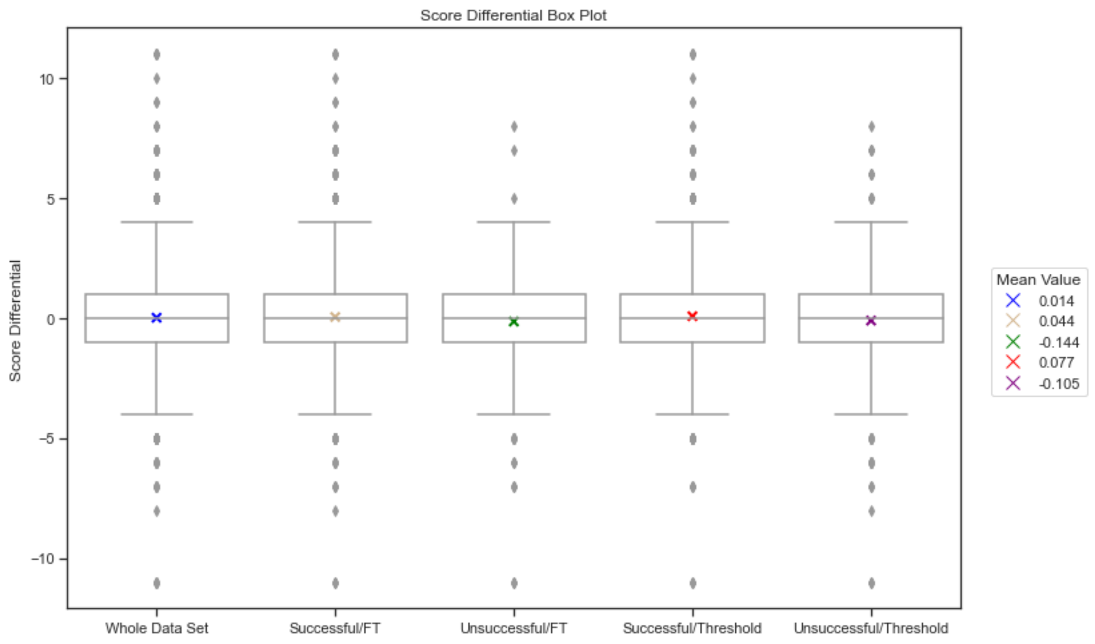

Figure A.9 – Score Difference Box Plot

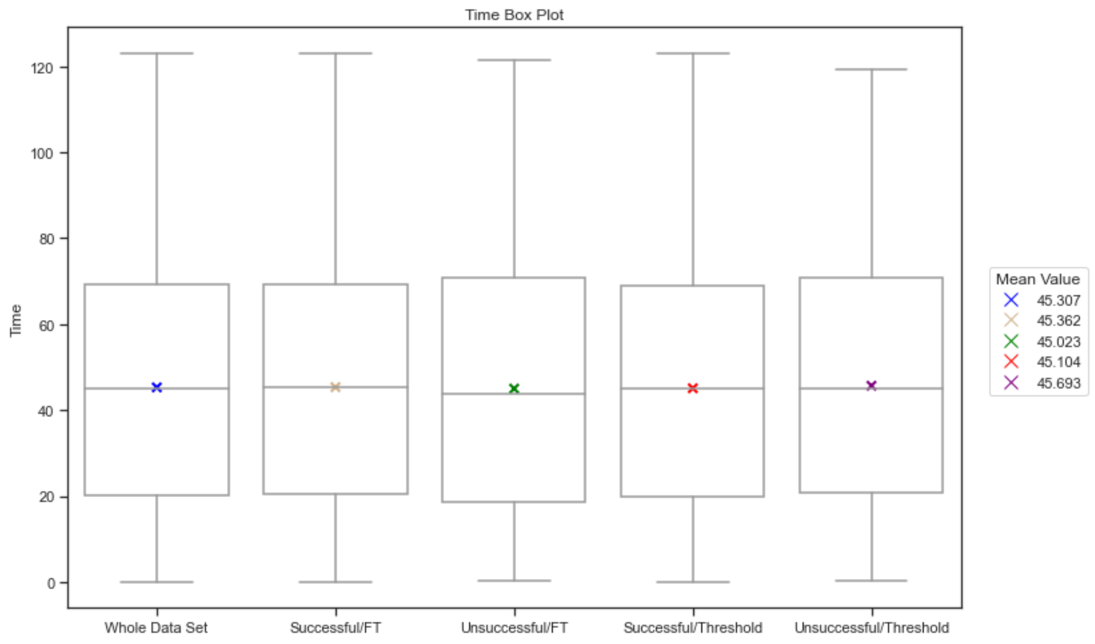

Figure A.10 – Time Box Plot

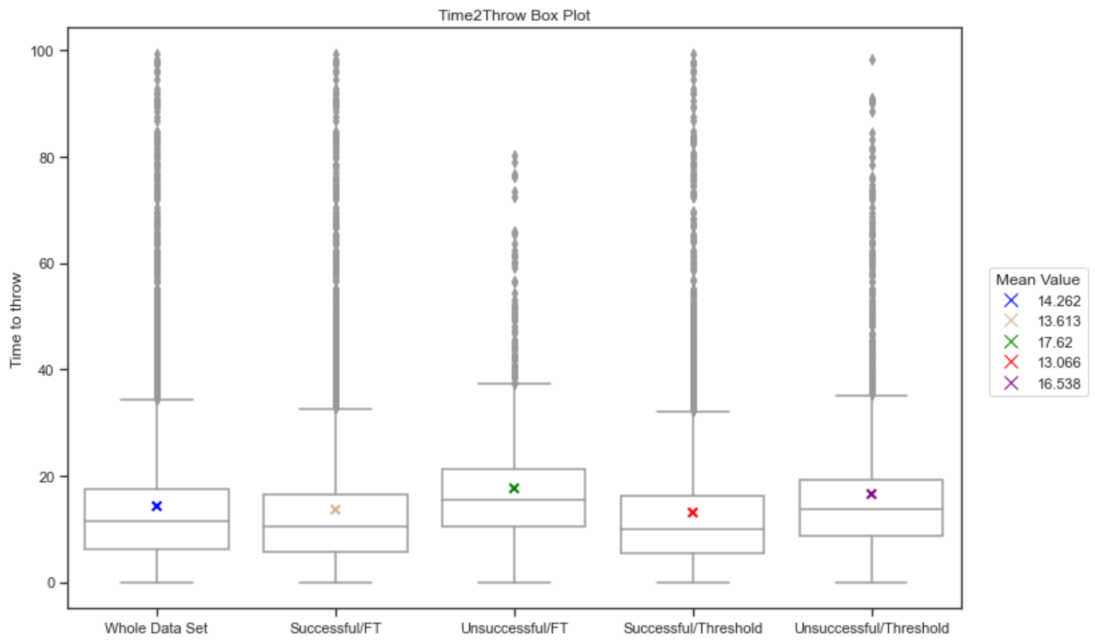

Figure A.11 – Time to Throw Box Plot

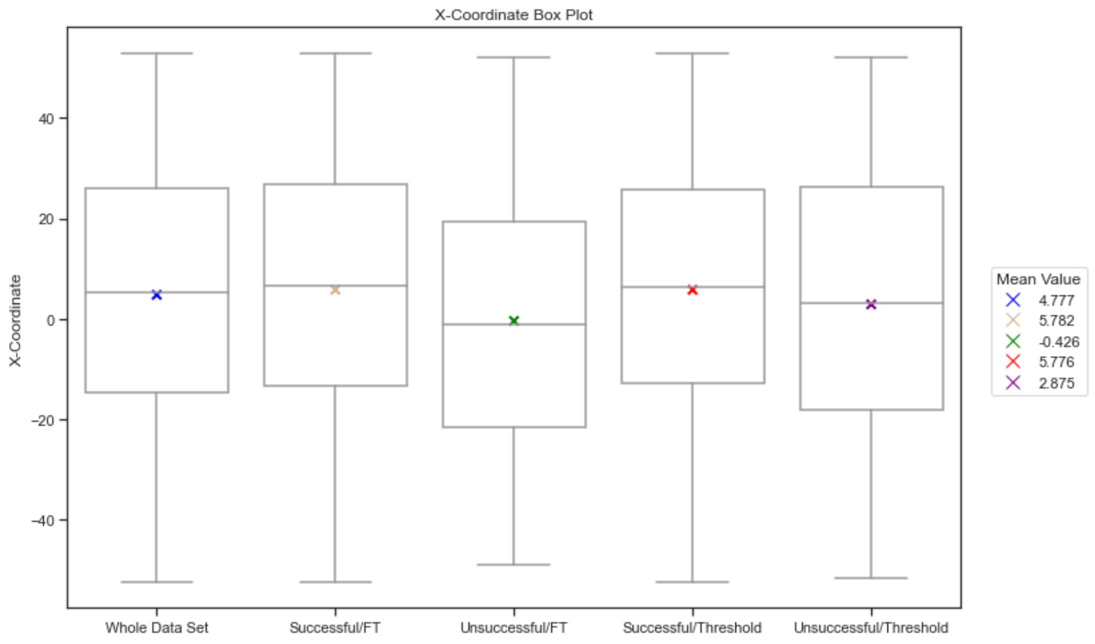

Figure A.12 – X-Coordinate Box Plot

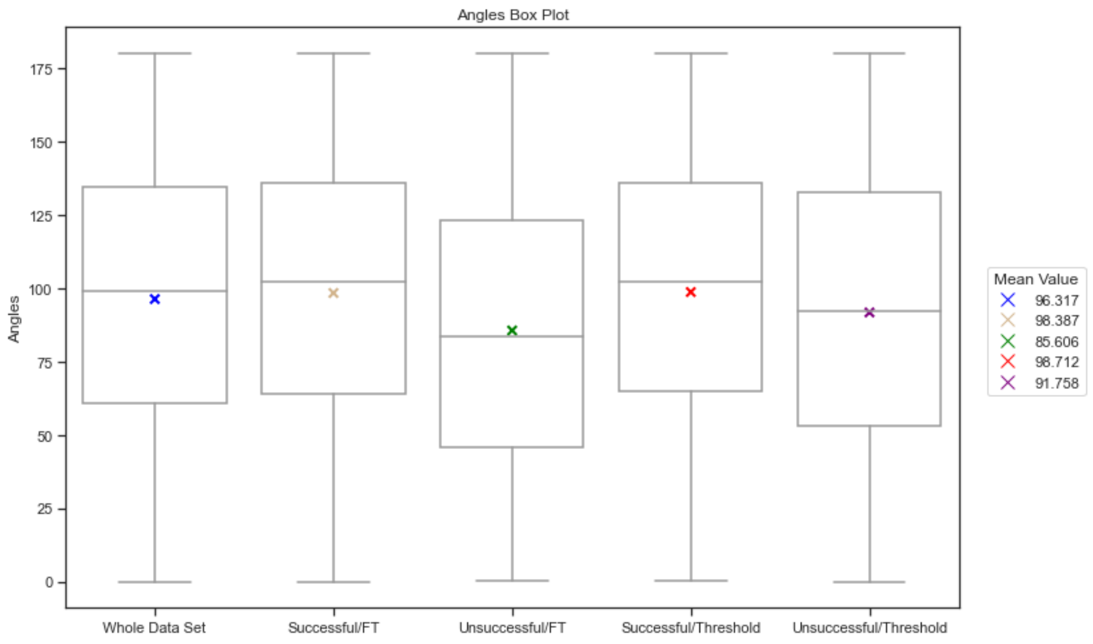

Figure A.13 – Angles Box Plot

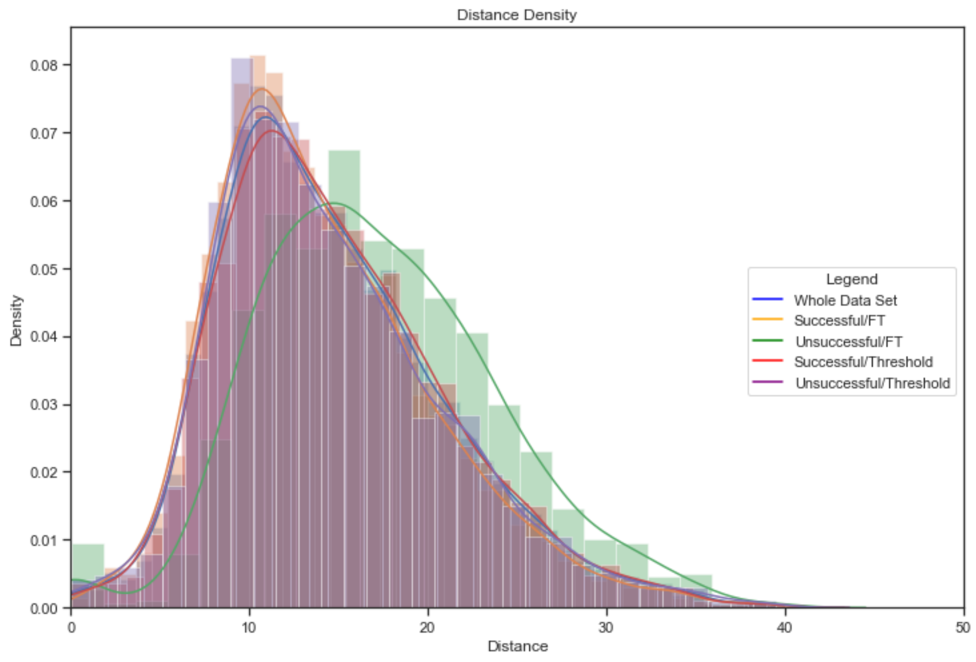

Figure A.14 – Distance Density Plot

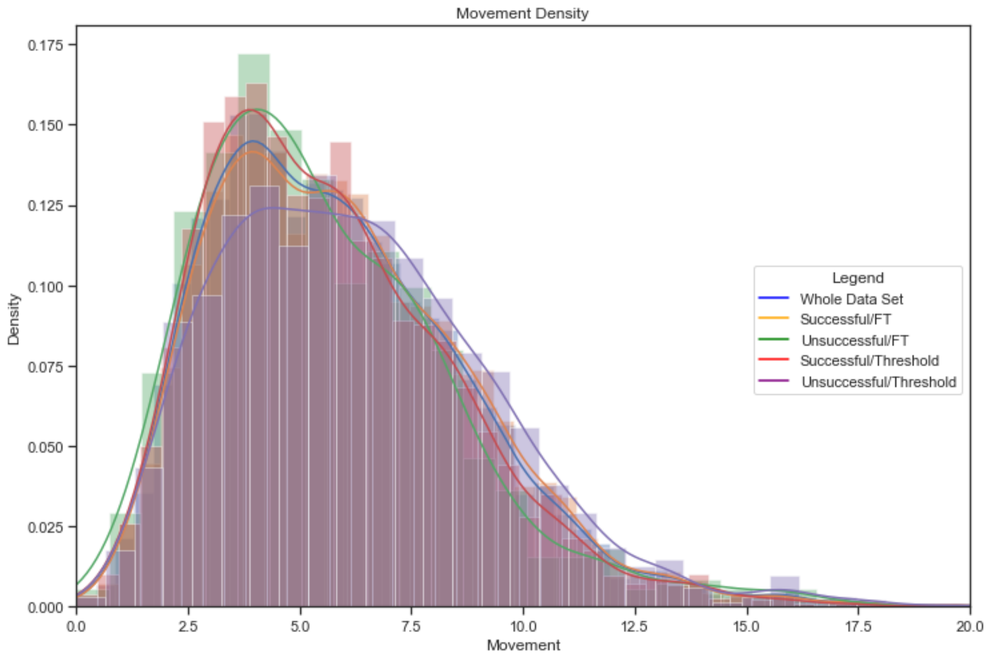

Figure A.15 – Movement Density Plot

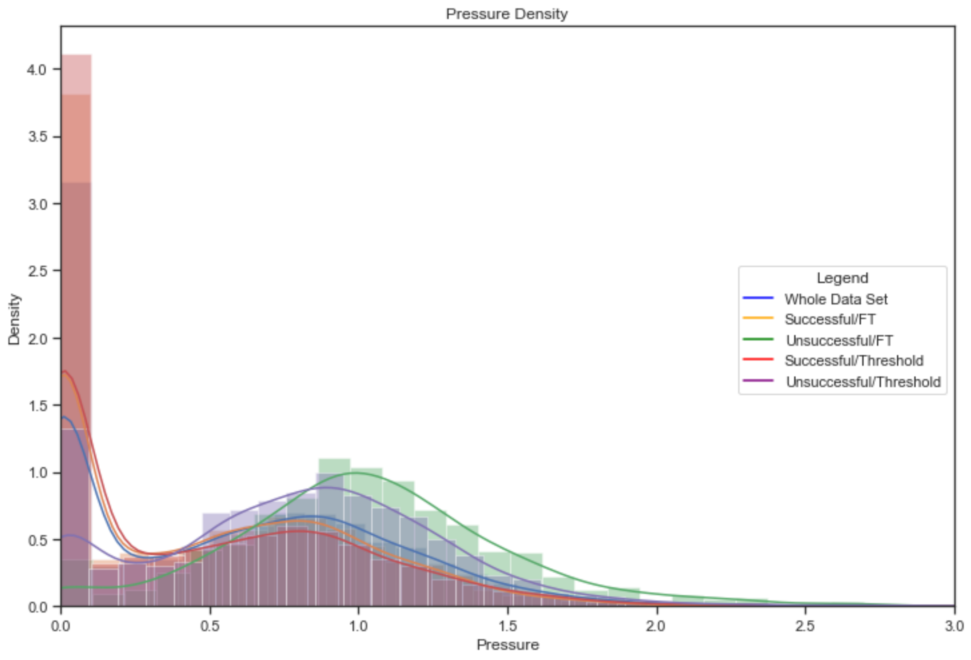

Figure A.16 – Pressure Density Plot

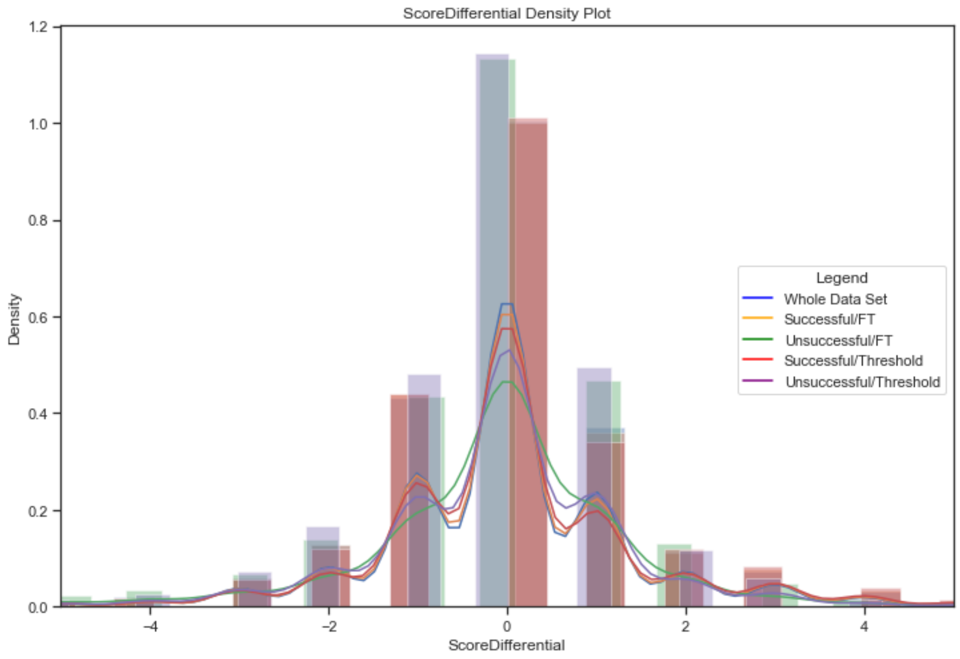

Figure A.17 – Score Difference Density Plot

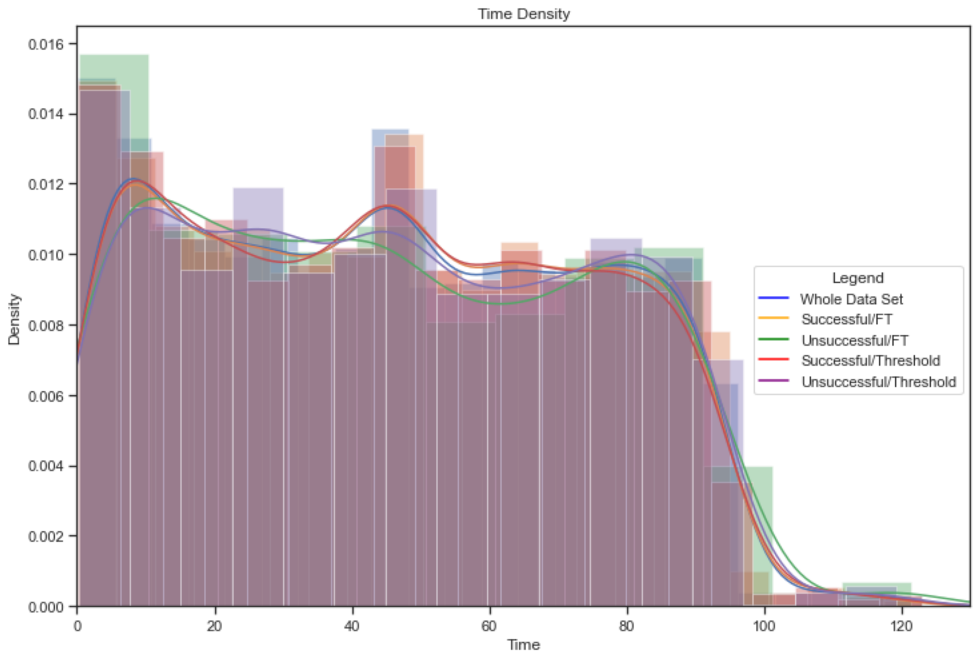

Figure A.18 – Time Density Plot

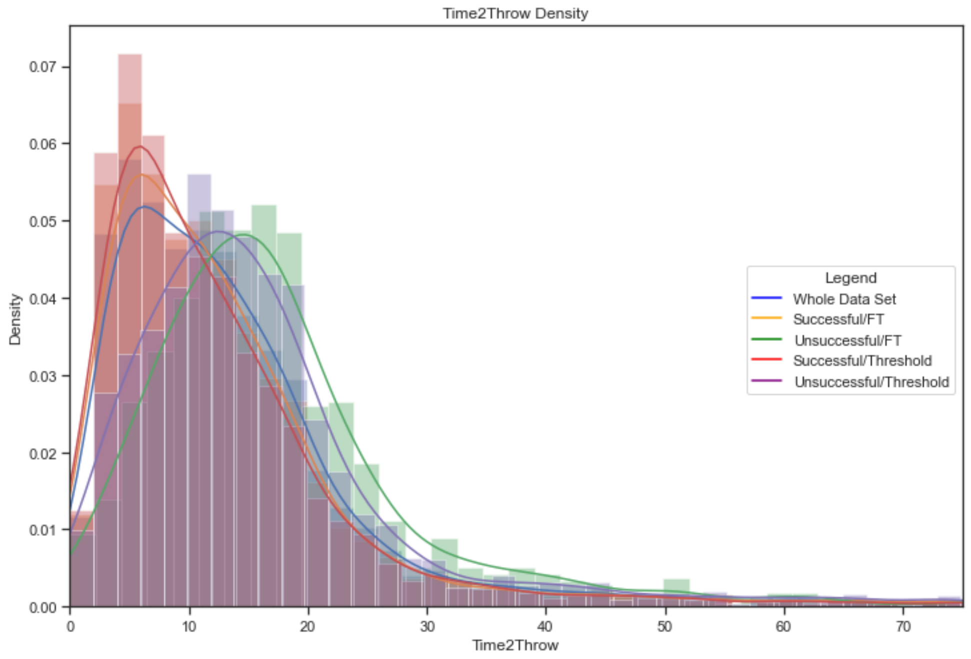

Figure A.19 – Time to Throw Density Plot

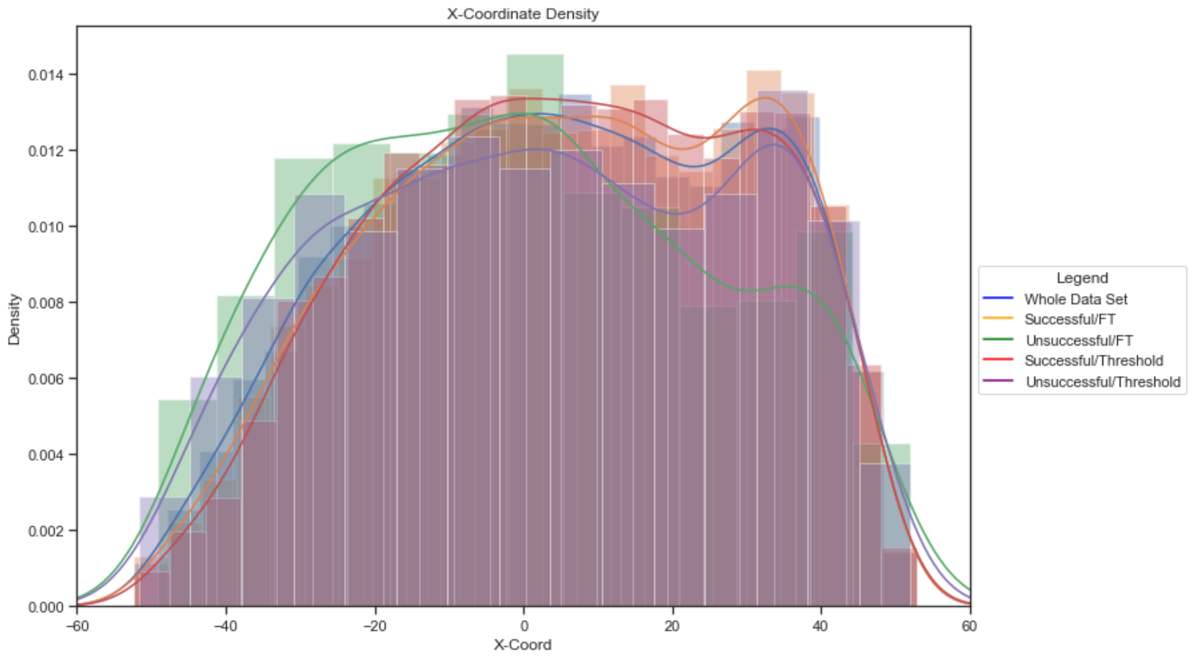

Figure A.20 – X-Coordinate Density Plot

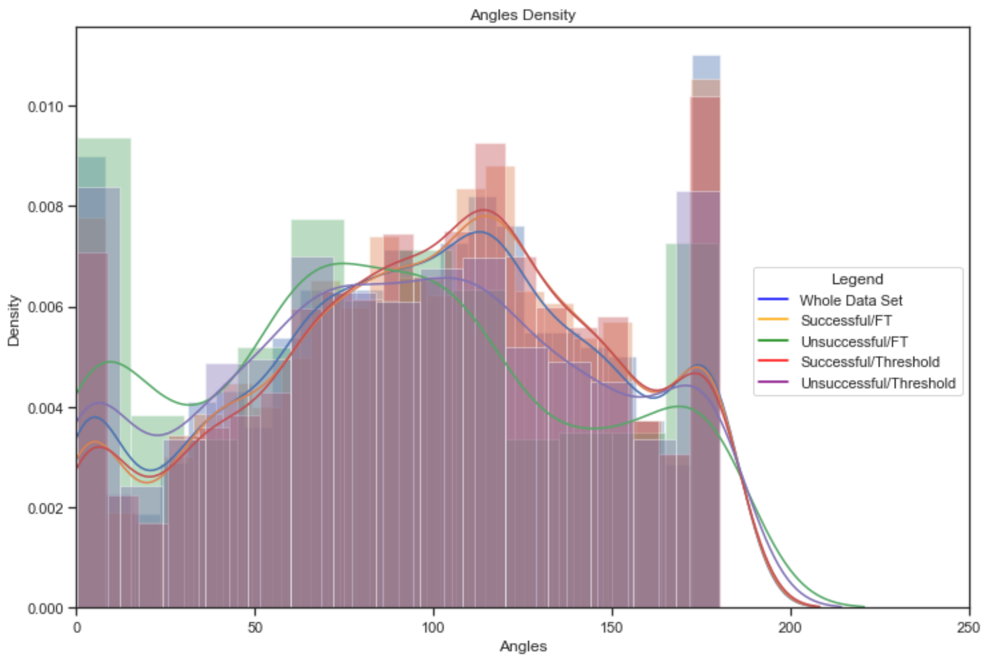

Figure A.21 – Angles Plot

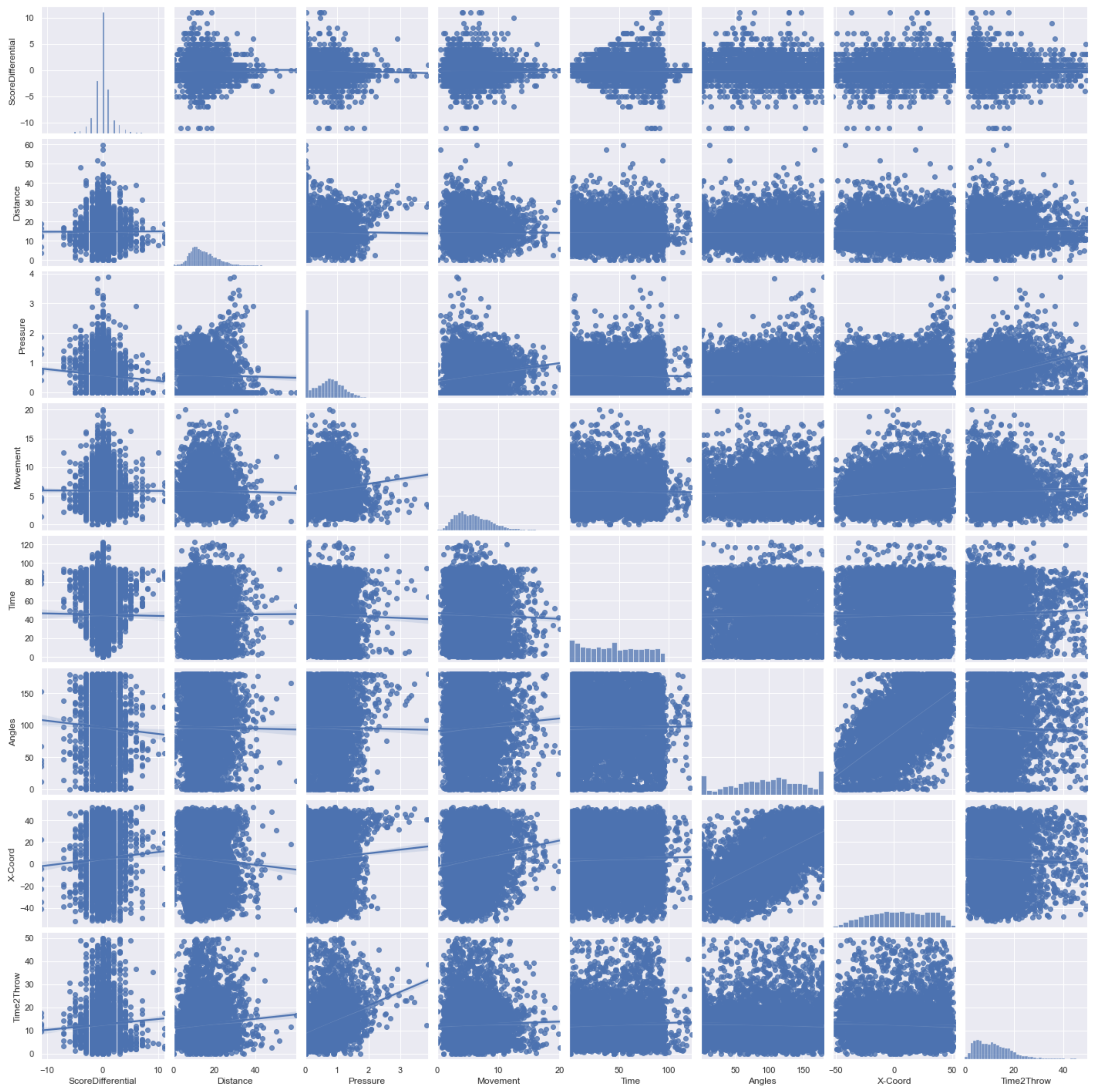

Figure A.22 – Scatter Plot of Feature Combinations