Universiteit Leiden

ICT in Business and the Public Sector

Pass Outcome Prediction in Football

Name: Jonathan Robijn
Student-no: s1045156

Date: 02/08/1991

1st supervisor: Arie-Willem de Leeuw
2nd supervisor: Arno Knobbe

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In football, the outcome of a match greatly depends on the actions of, and the interaction between, individual players taking part in the match. One of the most common interactions between teammates during a football match is the pass, during which one player sends the ball to a teammate in the hope of creating opportunities for scoring goals and avoiding the risk of opponents gaining ball possession.

In this work, we will investigate which factors have the most influence on pass outcomes and we will demonstrate how this information can be transformed into valuable information for potential users. For this purpose, existing state of the art position data and event data for 6 tournament matches is used to generate a dataset of thousands of passes. Extensive feature engineering and statistical analysis will be used to build numerical descriptors of the many facets of a pass. Following this, logistic regression will be used to train models that predict pass successfulness and we will analyze the importance of the features in the models.

We found that features describing the spatial relation between opponents, the pass vector and the pass receiver are the most influential for pass outcome. A probabilistic predictive model for pass outcome performs significantly better than the naive baseline model, with an accuracy of 74.4% (vs 68.1% baseline) and a precision of 88% on positive pass outcomes (vs 68% baseline). Additionally, several use cases for the probabilistic model output are evaluated, such as quantifying pass risk for the purpose of player decision making analysis and labelling passes with probabilities for the purpose of data mining pass outliers for post-match analysis. With our research, we obtained a deeper understanding of pass successfulness in football and we can provide usefull insights for practical users.

Contents

Chapter 1

Introduction

The global sports marked was worth nearly $488.5 billion in 2018 and is expected to grow at an accelerated rate to over $614 billion in 2022 [16]. Part of this growth is attributed to technological changes - specifically an increase in the number of Internet accessible devices. Unsurprisingly, the global sports technology market is growing fast as well, with an expected growth from $8.9 billion in 2018 to $31.1 billion in 2024, across a wide variety of sports ranging from baseball to golf [33].

Of all these sports, football is the most popular sport in the world, being played by 250 million players in over 200 countries. It should come as no suprise that it is also the sport with the biggest market share - the European football market alone is worth an estimated € 28.4 billion [25]. The largest segment in the global sports technology market is directly linked to football, where technology is increasingly used to support decision making through descriptive and predictive analytics [33]. Needless to say, there is a very real demand for novel data analysis methods within this domain.

Nowadays, there is a variety of data being generated during both training and matches in high-level football. Players and coaches have access to increasingly more detailed information about player performance based on data originating from sources such as accelerometers, heart rate monitors and time motion analysis [1]. In this work, we will focus on a specific form of time motion data - real-time position data and event data.

Position data provides the coordinates of all participants on the field at a certain frequency for the entire match or training session. This gives us a detailed picture of the whereabouts of all match participants at any point in time. Meanwhile, event data provides rich contextual information that describes exactly what was happening at specific moments in terms of player actions (e.g. goal shots) and general match events (e.g. interruptions). The two types of data are closely linked, and can be used to enrich each other. However, with all this information being available it can be a daunting task to figure out where to look for valuable insights for a coach or a player.

Tactical analysis typically focuses its attention on the events that occur during a football match, including their outcomes and spatiotemporal contexts. Of these events, the *pass* is the most common one and it involves one player, who is in possession of the ball, attempting to pass the ball to another player on his team. In an average football match hundreds of passes are performed. Passes have a huge impact on the flow of the game, creating opportunities for shots on the goal and risking a loss of ball possession in case of failure. In the literature, aggregate pass metrics such as pass completion rate and number of passes are considered important indicators for how well a team performs in a match [37][15].

In this thesis, we will focus on the analysis of the outcomes of pass events - whether the passed ball actually arrives at the intended receiver. In particular, we will focus on identifying which spatiotemporal characteristics of a pass are the strongest predictors for pass success and we construct predictive models for pass successfulness. The questions we will set out to answer are "What are the most important factors determining pass success in a football match?" and "How does a predictive model built on these factors add value for the stakeholders involved?". In order to work towards this goal, the project will take the shape of a typical machine learning project with a familiar pipeline ranging from data cleaning to model assessment.

In the next chapter, the background literature will be discussed, covering previous work done within the domain of football data analytics and more specifically spatiotemporal analytics. In chapter 3, the methodology used in this project is discussed, primarily covering the predictive modelling and model assessment work. The construction of the initial pass dataset and the feature engineering are then covered in chapter 4 and 5, as they both require a significant amount of domain-specific work to be done. Finally, the results are highlighted in chapter 6 and their implications are discussed in chapter 7. We then discuss interesting use cases in chapter 8. Finally, the thesis is finished with a conclusion in chapter 9.

Chapter 2

Related work

Until a few years ago, the detailed spatiotemporal position data that is currently available was scarce and very costly to acquire. This made research into the tactical behaviour of teams and their players very difficult to do using this type of data. The primary type of data available then was hand-annotated event data, typically labelled and gathered during a live football match by human annotators. This type of data mostly consists of events involving either the ball or special match interruptions. While this data has obvious limitations, only describing player interactions with the ball, it was still used extensively in research.

The authors of Ref. [30] used ball event data (referred to as partial team tracings) to infer team behaviour. Their work allowed them to identify general movement patterns of the entire team, which in turn enabled them to correctly identify teams with a known similar playstyle and differentiate between teams with different playstyles.

Similarly, the authors of Ref. [32] used the same kind of ball event data to analyze team strategy and behaviour. They interpolated the positions of the ball and possessing players in between individual ball events to approximate their locations at any particular moment. These interpolated values are then used to determine how the ball travelled during the period in which a team has ball possession and this is used as a proxy for team strategy and behaviour. The authors used this approach to highlight factors explaining why teams playing at home tend to perform better than the away teams.

While the work mentioned so far focused on interpolating player positions from the individual ball events, a lot of interest has also been shown in the analysis of passes. For example, recurring pass sequences can be extracted and transformed into insights about the passing patterns of specific teams [26]. From these sequences, the strategies a team tends to employ for specific goals, such as maintaining ball possession or entering the attacking third, can be derived.

Finally, the authors of Ref. [10] used ball pass events to build a predictive model that predicts whether a shot (on goal) event will happen within a certain period of time after the pass is completed. This model was then used to assign value to individual passes, based on the start and end positions. Professional football players were valued based on the values of the passes they completed and the authors found that some of the most well-known players were rated very highly by the model. Additionally, the features of the trained model were used to determine which areas of the football field are the most valuable placed to pass the ball from and to [10].

The overarching theme of most research done with hand-annotated ball event data is the movement of the ball and the way the match participants interact with it. Any

other information was usually interpolated based on the labelled event data and their associated field positions. This puts a clear limit on how much accurate contextual information is available.

2.1 Spatiotemporal position data analysis

Due to technological improvements and the more widespread distribution of tracking systems, spatiotemporal position data has become more widely available. Previously, this type of data was mostly available through specific vendors, while every major professional football club now seems to have some form of position tracking system available on their fields [1]. The generated data describes the position of all match participants and the ball in a precise and continuous fashion.

On its own, position data already has many use cases such as training load monitoring [1]. Combined with heart rate monitors, position tracking is used to monitor the effort exerted by players during a training. The main goals for this are improving player performance and preventing injuries [1].

For the tactical analysis of matches, position data also offers an enormous amount of information. The volume of data is so high that descriptive statistics is typically employed to break the data down to several aggregated values that can be interpreted more easily. The same principle was applied in previous work, such as the work done in Ref. [34], where the authors used multivariate techniques to break match statistics up into several principal components. These components were then used to cluster teams into groups and it was found that the clusters tended to be more closely associated with either winning or losing teams. By extension the authors figured out that metrics such as "shots on goal" and "percentage of ball possession" can be used to disciminate the winning teams from the others. [34].

For position data specifically, breaking the per-player positional information down into several aggregate metrics shows promise for summarizing team dynamics. Preliminary analysis has shown that metrics such as team centroids and team surface areas are very descriptive for goal-scoring opportunities [21].

These are some of the ways in which all the available position data can be broken down into performance metrics covering the entire match. However, combining position data with ball event data provides us with another way to analyze tactical behaviour of players. While the ball event data describes *what* happened, the position data describes *how* it happened. The combination of the 2 types of data allows us to focus our analysis on specific moments of interest in football matches, and the spatiotemporal data describing the situation around those moments.

This naturally brings us to the analysis of match events, such as shots and passes. While previous work was mostly limited to analyzing the events themselves and/or chains of events, the inclusion of spatiotemporal position data allows us to include more detailed contextual information, such as how players of both teams interact with the event.

As an example, the authors of Ref. [31] evaluated the 10 second time window preceding a shot on target and created a model that predicts whether the shot on goal will actually result in a goal. A dataset containing nearly 10000 shots was extracted from available event data. The associated position data was then used to generate predictors relating to various types of contextual information, such as the defender proximity, speed of play and the origin location of the shot. The final trained model is a regression model that predicts an Expected Goal Value which can approximate the total number of goals

scored over larger collections of shots. The model performs reasonably well, with the Expected Goal Value typically being quite close to the actual number of shots. The authors also found that spatial features such as defender proximity and interaction with surrounding players were very influential for the final predictions [31].

2.2 Pass analysis

The greater part of the work done on the tactical analysis of match events, using spatiotemporal data, has been focused on passing. Previous work had already been done using just ball-event data [10][26][49]. Focus was then mostly on the players directly involved in the pass, and any information outside of pass events had to be interpolated.

With the greater availability of spatiotemporal position data, new work has been produced which focuses on a deeper analysis of match events, including information on inter-player interactions and other contextual data. The work on this subject can be divided into 2 main areas of interest - pass reward and pass risk.

Pass reward describes the value or quality of a pass - how beneficial successful execution of a pass is. The brunt of the work on the tactical analysis of passing has been focused on this particular aspect.

The authors of Ref. [13] used a combination of event data and position data to train a predictive classification model that can be used to classify passes according to subjective labels "Good", "OK" or "Bad". The model was trained on numerical predictors extracted from the position data for every pass in the event dataset. Their model had an accuracy of up to 85% and had a level of agreement with human observers that is very similar to the level of agreement between different human observers.

A classifications model for pass reward was trained in Ref. [41]. The authors extracted a pass dataset from ball event data and then performed feature engineering to extract predictors from the associated position data. Some of the factors included in the feature engineering are the location and velocity of the sender and the intended receiver of the pass. The proximities to the nearest opponents are also included. The trained models are used to classify whether a pass is effective (defined as leading to a shot on goal). Additionally, the authors used the probabilistic output of the trained models to provide advanced statistics describing the passing behaviour of teams and their players. These statistics can be used to help explain the outcome of matches in a more thorough way than is possible with basic passing metrics such as total number of passes and number of completed passes can.

Instead of directly valueing a pass based on its attributes, it is also possible to value said pass based on how it changes the situation on the field. The authors of Ref. [29] created a definition of "dangerosity" - how likely a player in ball possession is to score a goal at his/her current position. This definition of dangerosity was based on various factors such as the zone of the player and the pressure opponents are putting on said player. This definition of dangerosity was then aggregated over the entire match and it was found that dominance - the difference between aggregated dangerosity of both teams, had a much higher correlation with match outcome than any naive metric such as shots on goal. This promising definition of dangerosity can also be used to value passes based on the change in dangerosity they cause [29].

All of the pass reward models mentioned so far either have no quantitative output or are trained based on the likelihood of rare events like shots and goals happening after the pass. This means that the quantitative models mostly overvalue forward passes. The

authors of Ref. [23] instead trained a quantitative model based on a new definition of "defensive disruptiveness". This model instead values passes based on how much they disrupt the defensive formation of the opposing team, through metrics such as the change in displacement and spread of the opposing team. The authors found that their newly defined defensive disruptiveness score can be appoximated well with a linear regression model based on simple pass characteristics. The D-Def score provides an interesting alternative method for evaluating passes that doesn't overvalue forward passes due to a reliance on shot or goal events [23].

All of the work mentioned so far values passes based on pass reward - the potential impact successful execution has. The other metric that can be used to value passes is that of pass risk, the likelihood of the pass being executed successfully in the first place.

In Ref. [41] a pass risk model was trained alongside the pass reward model. The same features, such as locations and velocities of involved players, were used in both models. However, the pass risk model was instead trained on classes describing whether the pass successfully reached the intended receiver. The authors used this trained pass risk model to provide advanced statistics on the overall riskiness of passes done by a player or team.

The authors of Ref. [49] trained a regression model which assigns probabilities of pass success to passes. They did this using only ball event data and implementing proxies for opposing players' pressure on the pass participants. Their model performed better than a simple baseline model which always predicts the proportion of pass success of all passes done during the previous season worth of football data.

2.3 Positioning of our work

In the literature on football pass analysis significant work has been done in the area of valueing pass reward. As was shown previously, comparatively little work has been done on valueing pass risk.

The work that does investigate pass risk models trained said model on the same features as the pass reward model, with the only difference being the predicted variable [41]. Additionally, the performance of the final model and the impact the different categories of features have on the model's performance were only briefly mentioned in terms of log loss values [41], which are not easily interpretable.

The only work that specifically focused on building dedicated pass risk models only had ball event data available [49]. Obviously, the lack of spatiotemporal position data restricts the level of detail that can be captured in any trained models.

In our work, we perform a detailed analysis of dedicated pass risk models trained on a combination of ball event data and position data. Additionally, we describe in detail how much every feature we extract from the data contributes to the performance of the final models. The goal is to provide a useful future reference for spatiotemporal feature importances for pass risk.

Chapter 3

Methodology

The goal of this research project is to build a probabilistic predictive model that can be used to predict the likelihood of a pass succeeding. As we start off this project, we have a collection of data describing all the events and player positions in a number of matches in a tournament. To get from this state to the eventual goal, a lot of work is required in the realm of data cleaning and machine learning.

3.1 Overview of the pipeline

Initially, we have several football data sources available to us that each describe certain aspects of several footbal matches. While it is very rich data, it is not yet in a structured format that will allow any machine learning model to be trained on it. To get from this state to a trained model that can handle pass predictions a lot of work is required. This work can be split up into several parts that together form a pipeline of operations. A rough outline of the main steps in this pipeline has been illustrated in figure 3.1 and we will now give a high level overview of the input and output of each step.

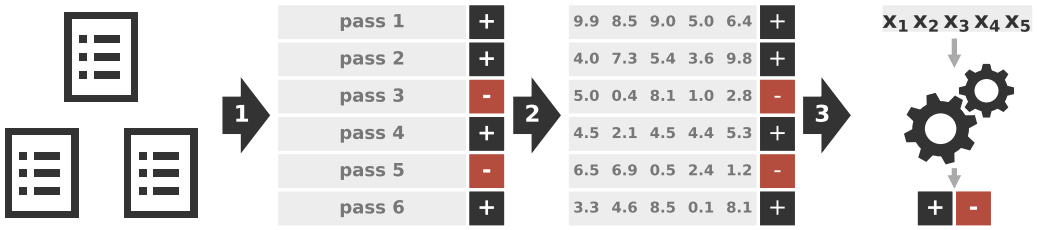


Figure 3.1: The data science pipeline of this research project. The main steps are (1) Data transformation, which creates a structured pass dataset out of existing data sources, (2) Feature engineering, which enriches the dataset with numeric pass properties that can be used for machine learning and (3) Modelling, which trains a predictive model

Data transformation In this first step, we will take the collection of data that we have available and transform it into a structured dataset describing a collection of passes and their outcomes. This dataset will be used as the foundation for the follow-up work on enriching the data with new features and building predictive models. Due to the domain specific work involved in this step, it is extensively described in chapter 4 together with a detailed description of the available data.

Feature engineering After the data has been put in a structured state where it clearly describes the collection of passes, we will move towards enriching this pass dataset with numeric properties that describe and distinguish the individual passes in a clear way. The goal is to end up with a pass dataset with plenty of numeric descriptors that allow us to find patterns identifying unsuccessful and successful passes. This step is the majority of the work and it is described in detail in chapter 5.

Modelling After we have created a structured dataset with numeric predictors, we use this dataset to train a model and assess its performance. The final output of this step will be a predictive classification model which takes the numerical features engineered in the previous step as input and provides a pass outcome prediction as output. The methods required to do this step, as well as the choices that have to be made, are described in the remaining sections of this chapter.

3.2 Feature importance assessment

In order to assess how useful specific features are for the purpose of separating negative from positive observations, we perform statistical tests to determine whether the distributions of the two classes are significantly different for each feature. To do this we perform a Kolmogorov-Smirnov test [44] for the continuous features. Since the KS test is not valid for categorical features we will perform a Chi-squared test for said features. Both statistical tests give us a *p*-value and for both tests the null hypothesis is that the two distributions are sampled from the same true distribution. If the *p*-value is below 0.05, we reject the null hypothesis and we consider the difference between the two distributions to be statistically significant.

Equally important is assessing whether features partially explain each other. If features are collinear with one another, a feature may be linearly predicted based on the others. When collinearity impacts the dataset, the results from the modelling step can become unpredictable and unreliable. After all, if two features describe highly related factors, the model could assign different combinations of weights to them and still get similar performance. One popular method for assessing dataset collinearity is the Variance Inflation Factor (VIF). This metric can be calculated for every feature in the dataset and tells us how little variance of the feature is left unexplained by the other features in the dataset. The formula is as follows:

$$\text{VIF}_i = \frac{1}{1 - R_i^2}$$

where VIF_i is the variance inflation factor for feature i and R_i^2 is the coefficient of determination for a regression model trained on the remaining features and an intercept term. As this formula shows, higher values mean there is less unexplained variance in

the feature in question. A popular rule of thumb is that if the VIF exceeds 5, the feature is highly collinear and can negatively impact the model's interpretability.

3.3 Categorical feature encoding

Most machine learning algorithms expect numerical input features. In order to include categorical input features, multiple methods are available to convert these features into numerical ones. The most simple methods are binary encoding and one hot encoding, and both convert categorical features into a series of boolean integer features.

In the case of binary encoding, the different categories in a feature are assigned integer values and these integer values are converted to a binary format. The number of numerical columns added is then equal to the longest binary format needed to encompass all the category integers. Alternatively, one hot encoding creates a boolean integer feature for every category in our categorical feature, where only one boolean integer feature can be activated at a time. This is a very simple and robust procedure that will allow any regression models to correctly interpret a certain category as being "active" for a particular pass.

Which of the two options is preferable is mostly dependent on the number of categories a feature has. With a large amount of categories, one hot encoding introduces sparseness into the feature matrix, while binary encoding minimizes the number of columns added. On the other hand, binary encoding introduces a dependency between the generated numerical features and if the number of categories is limited, this may be an unnecessary limitation. These two methods and several other encodings are also described and compared by the authors of Ref. [40].

3.4 Normalization

A lot of machine learning algorithms expect all input features to be on the same scale. For example, regularized regression penalizes high weight values and this makes it important that the weights of different features become comparable so the regularization does not bias the model in favor of certain features. If one feature is measured in metres and the other feature is measured in seconds with wildly different orders of magnitude, one of the 2 features may need much higher weights to be equally important in the model. Since regularization penalizes high weights this feature will not be able to be considered as important as the other feature in the final model.

In order to put features on the same scale, we will standardize the input feature values by subtracting the mean of the feature over the entire *training* dataset, centering the feature around 0, and then dividing the feature by its standard deviation over the entire *training* dataset, expressing each value in the dataset as a z-score describing how many standard deviations from the feature's mean the value of an observation is. The formula is as follows:

$$x' = \frac{x - \bar{x}}{\sigma_x}$$

where x is the feature value, $\bar{x}$ is the mean of the feature over the training set and σ_x is the standard deviation of the feature over the training set.

3.5 Dataset balancing

It is very common that not all classes are equally represented in datasets used for classification. Working with unbalanced datasets brings its own challenges. Models trained on these datasets tend to default to the majority class in a lot of fuzzy cases, since the models learns that one of the two classes in a binary classification problem is in fact the majority class. As a result, the model is biased towards said majority class.

There are a number of ways to rebalance an unbalanced dataset. If enough data is available, the majority class can have its size reduced. This process is called downsampling and is typically done through a semi-randomized process called random downsampling, in order to avoid bias. If the amount of available data is not enough to allow the removal of observations, the alternative is to instead increase the size of the minority class. The most common approach to doing so is through a semi-randomized process of duplicating observations from the minority class, called random upsampling. However, more advanced methods like SMOTE [12] are available which generate artificial observations by interpolating between multiple observations of the minority class.

The authors of Ref. [4] evaluated several methods on a variety of datasets, ranging from random downsampling and random upsampling to SMOTE. The authors found that while more advanced methods like SMOTE perform well on datasets with a small number of positive observations, random upsampling tends to perform well on datasets with a larger number of positive observations.

3.6 Model choice

There are hundreds of classification algorithms available, ranging from simple to highly complex. The right choice of algorithm depends on the type of input data we will have.

Neural networks (NN) are very powerful and can find patterns in very complex input data, being used in a variety of fields ranging from image object detection [52] to natural language processing [51]. Neural networks are capable of modelling incredibly complex relations due to the potential size of the neural net. Given enough data and time to train a neural network is thus likely to give the best possible performance. However, they are difficult to interpret [22] and time consuming to train while similar results might be achievable with much simpler algorithms, depending on the type of input data and size of the dataset. NNs also can't handle categorical features in their raw format.

Random forests (RF) are an ensemble based method that relies on generating many individual decision trees that each output a prediction [19]. The Random forest then aggregates the output of the decision trees into one aggregated prediction. Due to the nature of decision trees, RFs can handle multi-class classification problems naturally and handle categorical feature data without any extra work. Additionally, the underlying decision trees can be used to output probabilities for predictions, based on the class proportions in the leaves of the trees. However, these probabilities are limited and extra work is required to calibrate them [36].

Support vector machines (SVM) are a maximum margin based classifaction method that relies on optimizing the mathematical distance between observations of opposite classes [20]. Various kernels are available to convert the problem space to one where observations can be more easily separated. This means it is inherently based on calculating the distance between numerical features and extra work needs to be put into

converting categorical features. SVMs are known to scale well with higher amounts of features, being very well suited for handling sparse data.

Logistic regression is a relatively simple and well understood algorithm [18][8] that has been applied in a variety of domains [38]. It is an extension of linear regression [6] that is based on optimizing a likelihood of specific observations belonging to specific classes. Due to this it is also one of the few algorithms capable of useful naturally probabilistic output without additional work [36]. Just like SVMs and NNs, logistic regression can only handle numeric features.

The authors of Ref. [3] compared the classification performance of logistic regression to that of support vector machines in a number of experiments and it was found the performance was very similar, apart from SVMs performing slightly better for very unbalanced datasets. The authors of Ref. [50] compared logistic regression to neural networks for performance on medical data and while neural networks tended to perform slightly better, their results were much less interpretable and much more time consuming to get. They also found that the predictions from the logistic regression model were more robust. However, logistic regression was also compared to random forests and it was found that random forests performed better in the majority of cases [17].

Regularization is a technique that can optionally be used in order to prevent overfitting in logistic regression. 2 variants, L1 and L2 regularization, are available and each restrict feature weight magnitudes in a slightly different way, as described by the authors of Ref. [35]. L1-regularization tends to have the model converge to a state where weaker features' weights get reduced to 0 due to the nature of the absolute weight values. Meanwhile, L2-regularization will lower weaker features' weights significantly, but they will not be set to 0. In practice, this makes L1-regularization very useful as a mechanism for feature selection - weak features are essentially turned off automatically during model training.

3.7 Model hyperparameter optimization

Most machine learning algorithms have one or more hyperparameters that can be tweaked for optimal performance for a given problem. Simply optimizing our hyperparameters based on the trained model's performance on the test set biases our test score and makes it unsuitable for accurately assessing the generalization error on the greater set of all possible passes, as can be seen in Ref. [11]. It is clear there is a need for a validation set, disjoint from the existing training and test sets, which can be used to compare the performance of different hyperparameter configurations.

The authors of Ref. [43] evaluated a number of sampling methods that can be used for splitting datasets, such as Simple Random Sampling, Systematic Sampling, Stratified Sampling. Simple random sampling assigns an equal probability of being picked to each observation. It is the most efficient method while still giving similar results. Stratified Sampling is designed to make sure each class is represented fairly in every fold, when dealing with an extremely unbalanced dataset.

In order to find optimal hyperparameter values, we first need to determine which hyperparameter configurations we will test. There are several methods of doing this including exhaustive grid search [39], random search and more advanced methods. Random search is known to be more efficient and achieve similar or better results when a large number of hyperparameters are involved and the search space increases exponentially, as was shown by the authors of Ref. [5]. There are many more advanced

algorithms available, as described in Ref. [14], but if the number of hyperparameters is limited, simple exhaustive grid search can be an effective method for optimization.

3.8 Model performance assessment

While training a model is a fundamental part of any machine learning project, an equally important task is that of performance assessment, and doing so in a fair and representative manner. If model assessment is done incorrectly, the results may be unrealistic or simply incorrect. If this happens, the results may be wildly different when the model is used in practice and we would actually have no idea what our model's performance is going to be like in that scenario.

In general, it is better to collect a distribution of numbers rather than one single number in order to assess model performance. To achieve this, we use 10-fold cross-validation [27][42] for the train-test splitting while also using cross validation for the train-validation splitting. The result of this is a nested cross-validation procedure where an outer 10-fold cross-validation loop takes care of the train-test set splitting and model performance assessment while an inner 10-fold cross-validation loop takes care of the train-validation set splitting and hyperparameter optimization. Overall, a total of 110 $(10 \cdot 11)$ models are trained, making it quite a computationally expensive procedure. This needs to be taken into account for choosing a model.

Aside from looking at the average accuracy over all test folds, we will also look at the confusion matrix, precision and recall [47] of our model over all the test folds in our cross-validation procedure. This will give us insights into how well our model performs on the two pass outcome classes. Finally, the precision-recall curve will be drawn to evaluate how well our model performs with a changing threshold for the translation of probabilities to predictions. Receiver operating curves [7] are typically used for machine learning purposes, but the authors of Ref. [45] found that precision-recall curves tend to be more useful for assessing models on unbalanced datasets.

Chapter 4

Data exploration and transformation

Any machine learning project is only as good as its data. We used position and event data from the Amisco position tracking system (owned by STATS LLC) as the basis for our research project. This data contains detailed information about the exact locations of the players as well as contextual information describing what actually happened at a particular moment.

In this chapter, we will go over the technical details describing the position data, event data and player metadata we had access to, including their formats and contents. We will discuss what useful information is contained within the data and how we used the data to construct a new dataset containing all the passes described in the original data. Additionally, we'll also discuss how we calculated an intended receiver for passes that were not successful. The final dataset will form the basis for the feature engineering work that will follow in the next chapter.

4.1 Position data

One of the 2 main output datasets of the Amisco position tracking system is the position data. This data describes the 2-dimensional positions of every participant at every measured timepoint. In the data we have access to, the positions of all players, referees and the ball are measured at a frequency of 10 Hz.

Since the Amisco system only tracks player positions horizontally, the player positions are described as pairs of X and Y coordinates. The coordinate system used by the Amisco system has its center (meaning [0,0]) at the center of the field and is illustrated in figure 4.1. The X coordinate covers the length of the field, while the Y coordinate covers the width of the field. Both coordinates match the exact distance in meters, so the goal lines are located at X coordinates -52.5 and 52.5 while the side lines are located at Y coordinates -34.0 and 34 for a field size of 105 by 68 metres. One of the main benefits of this coordinate system is that it's very easy to mirror the field when needed by simply multiplying all coordinates by -1. This will be useful for certain specific scenarios where the team playing in the opposite direction is the team of interest, as will be seen in the Feature Engineering chapter.

For our research project, the position data had been split into separate files per match half, giving us 2 separate files for each of the 6 matches we had access to. The various columns and their values are described in appendix A.1. There is a row in the position data for every participant at every measured timepoint. Since the measurement frequency was 10 Hz and there were 2 teams of 11 players on the field as well as 3

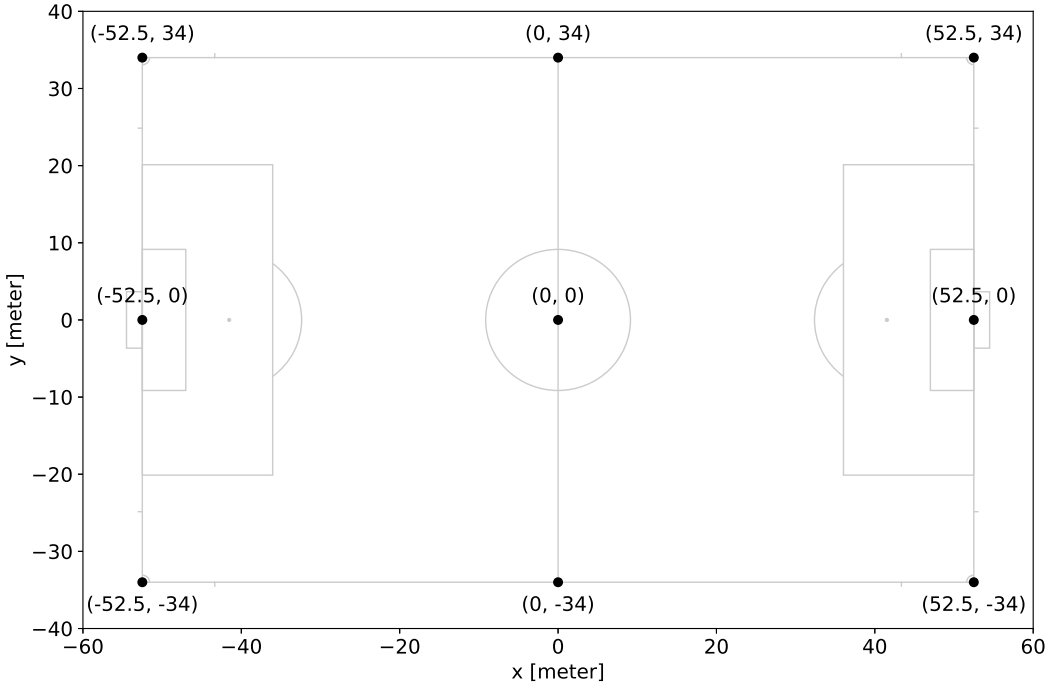


Figure 4.1: The coordinate system used in the tracking system. Several reference coordinates are shown in the format (x,y)

referees and the ball, this puts the total size of a file containing data for 1 match half at around 800000 rows.

4.2 Event data

The other main output dataset of the Amisco tracking system is the event data. While the position data describes what the exact positions of all participants were at any point during the match, the event data instead provides the contextual information that describes what exactly the participants were doing. This data is generated by human observers tagging events in real time, although there are some computerized algorithms involved that tag some of the more rudimentary events automatically. The human element does make this event data slightly more subject and error-prone, as will be seen later on.

Each row in the dataset marks an event that was registered with a detailed and well formalized description of how exactly it happened. For example, there is information about which parts of a player's body were used to perform the action. In general, there are 2 kinds of events that are actually stored in the event data:

- **Ball events** Actions that involve the ball and one or more players. For example: Passes or kicks. A table of all event types is listed in appendix A.4
- **Match events** Actions that interrupt the flow of the match in some way. For example: fouls or goals. See appendix A.5 for a table of all event types.

Events are always one of these 2 types of events, but never both at the same time. Each of the events is assigned an appropriate ball event code or match event code which describes what actually happened.

Similar to the position data, the event data has been split into separate files per match, once again giving us 2 separate files per match. The columns contained within the event data files are described in appendix A.2. It should be noted that the coordinates and timestamps in the event data are closely related to the values in the position data. Indeed, the event data seems to inherit its coordinates directly from the tracking system's position data. For ball events the coordinates match the ball coordinates in the position data at the same timestamp and for match events the coordinates match the position data coordinates of the primary player associated with that event, if any, at the same timestamp.

The event data largely focuses on events involving the ball, or events that interrupt the flow of the match for whatever reason. While that means the focal point of the event data is always the ball and any players directly around it and the rest of the participants on the field are mostly ignored, that is not a drawback for our research, since we are mostly interested in events directly involving the ball (passes). Each event data file containing data for 1 match half contains around 1000 ball- and match events, giving us an in-depth overview of what was happening around the ball at any particular moment as well as the intent of the involved players.

Now that we have a better understanding of the contents of the event data, we can use it to contextualize the position data. More specifically, we can use the event data to filter out interesting events and their associated timestamps, and then use the position data to derive the finer details pertaining to the participants' positions around that moment, looking only at position data that is actually relevant for the moments we're interested in.

4.3 Player metadata

Besides the 2 much bigger datasets we discussed before, there's one small, but not unimportant, dataset that we still need to discuss. The Amisco tracking system keeps track of the match participants as well as metadata describing their identities and roles within the match. This metadata is stored in what we're calling player metadata files and these files are essential for understanding the complete context of the scenarios encountered in the position and event data.

Unlike the other datasets, there's only 1 player metadata file per match, covering both match halves. All the participants in the initial lineup as well as any replacement participants placed on the field during the course of either of the 2 match halves are described in this one file.

While the event data tells us *what* happened and the position data tells us *how* it happened, the player metadata tells us *why* it happened and *who* was involved. It does this by linking the unique player identifiers used in the event data and position data to metadata records that describe details such as the identity of the participant and his team allegiance. In appendix A.3 a description is given of all the columns in the metadata. It should be noted that not all of these columns seem to be properly filled or used, holding just placeholder values. However, the table does give us an accurate picture of the team allegiance of the various players, as well as their identities and exactly what unique player identifiers they are linked to.

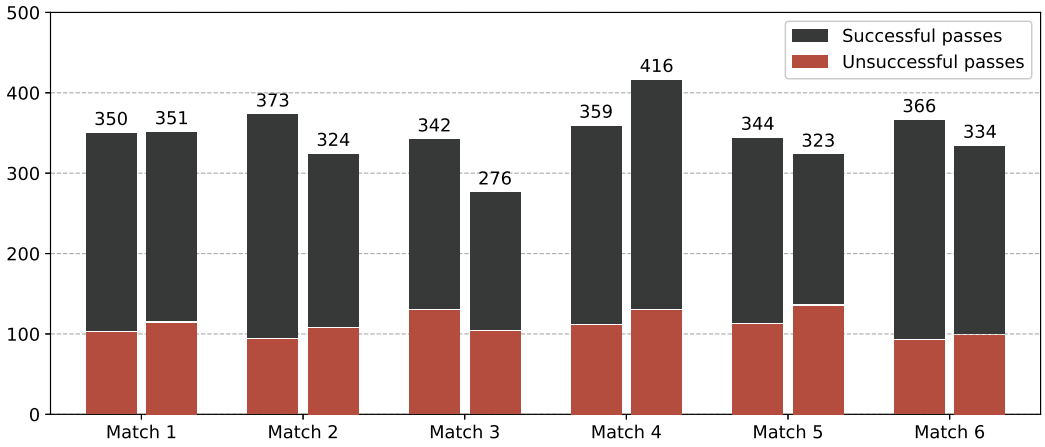


Figure 4.2: All **4158** passes in our dataset per match half

4.4 Constructing a pass dataset

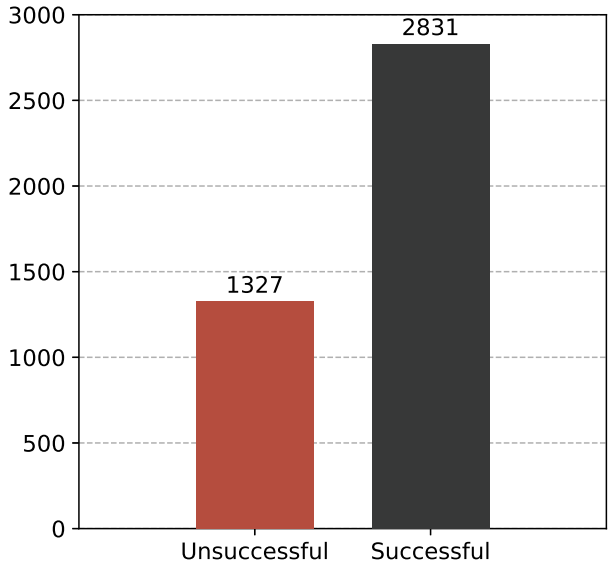
In this section, we will come back to the original goal of our research project, creating a predictive model specifically for passes. Now that we have a thorough understanding of the original data we have access to, we can finally leverage the data to create a new dataset optimized for the goals of our research. While the original data contains detailed information for every moment in the football match, we are only interested in a subset of these - the passes and their surrounding events.

By looking at only the rows with specific types of ball event codes, we can successfully identify the events marking the moment a ball was passed by some player. Finding the corresponding moment in which the pass was finished is relatively easy, as only events involving the ball or match interruptions are actually registered. In general one can look at the follow-up event right after any pass event to figure out which participant next exerted influence on the ball, and where and when that happened. Taking these pairs of events gives us a detailed description of when, where and by whom the ball was passed and when, where and by whom the ball was received.

If the next event is a match event, and not a ball event, a match interruption happened before the pass was received by anyone, in which case we consider the pass to have failed and mark the position at which the match event occured as the end location of the pass. If said match event was an interruption due to the pass being considered offside by the line referees, the match event usually has its location set to the position of the pass sender, making the pass unusable - we drop these edge cases.

For normal uninterrupted passes, we determine the pass successfulness by looking at the identity of the pass sender and receiver. If the 2 players belong to the same team (according to the player metadata) we consider the pass to have been successful and if the 2 players do not belong to the same team we consider the pass to have failed. Underlying this is the assumption that if the sender and receiver of the pass are of the same team, the receiver was indeed the intended one - we make this assumption because it is exceedingly difficult to check the sender's intent.

As it turns out, passes in the Amisco event data do not just include what domain experts typically think of as passes, those done during open play moments, but also set play actions from 1 player to another, such as corners or throw-ins. The 2 settings are typically regarded as being very different, having very different influential parameters



	Count	Percentage
Unsuccessful	1327	31.92%
Successful	2831	68.08%
Total	4158	100%

Figure 4.3: Class distribution in the pass dataset

and requiring very different skills from the players involved. After some initial modelling done later on, we also noticed that the set play "passes" were overrepresented among the missclassified cases, suggesting that they are indeed very different even from the perspective of a machine learning model. As such, we decided to also filter out any passes that were done during set play moments and the final number of passes then dropped from 4875 to 4158.

The final number of the passes we managed to extract from the 6 matches worth of data we had access to is highlighted in figure 4.2 and the final class distribution of our pass dataset is highlighted in figure 4.3 and as can be seen there our pass data has a significant class imbalance towards the positive cases.

4.5 Calculating the intended receiver

The goal of this research project is to create a predictive model that can predict whether a pass to a specified intended receiver is going to succeed or not, using contextual information as input. This means that we need to make sure an intended receiver is specified for every pass in our dataset in order for the data to be usable for training and testing a predictive model.

For successful passes, we make the assumption that the actual receiver is also the intended one. Unsuccessful passes are more complicated. With the 2 dimensional data that we have, we can make very few assumptions about which other participant the pass sender intended to send the ball to. The ball could be intercepted by an opponent early in its path, or closer towards its destination. If the ball goes out of field due to an error on the sender's part, it can be even more difficult to trace what exactly the pass sender's intentions were.

However, the one thing we do know is the path travelled by the ball during the pass up until the moment it's received by someone or the moment a match interruption

happens, whichever comes first. We will henceforth refer to the path travelled by the ball as the pass vector. Using this pass vector, we can estimate which of the pass sender's team members was the most likely target of the pass. We will be using a similar formula as proposed by the authors of Ref. [41] in their work on estimating pass risk and reward. The numerators and denominators seem to be erroneously swapped around in the formula listed in their work, resulting in the wrong numerical behaviour. To correct this, our formula is a slightly modified variant.

Essentially, we want to assign each team member a score, based on the pass vector and the team members' positions at the start and end of the pass. As was done in Ref. [41], we use the distance of the team members to the ball as well as the angle between the pass vector and the vector defined by the position of the pass sender and the position of the team members. These 2 components are then combined into one 'membership' score M_p for a given team member that is defined as:

$$M_p = \frac{D_{min}}{D_p} * \frac{\alpha_{min}}{\alpha_p}$$

where D_p is the distance of the current team member to the ball, D_{min} is the distance of the closest team member to the ball, α_p is the angle between the vector from the sender to the current team member and the pass vector and α_{min} is the angle of the team member with the smallest angle to the pass vector.

For the team member's distance to the ball we use the positions of the participants and the ball as they were at the end of the pass, as highlighted in figure 4.4. For the angles between the sender-to-team-member vectors and the pass vector we instead use the participants' positions as they were at the beginning of the pass, as shown in figure 4.5. This way the former is used as a proxy for the end result of the pass whereas the latter is used as a proxy for the original intent of the pass sender and we include a trade-off between both factors in the formula. In the end, the team member with the highest membership score gets selected as the intended receiver for the unsuccessful pass in question. By doing so, we can use the unsuccessful passes as negative cases for machine learning purposes.

This section went through all the steps undertaken to construct a usable pass dataset. Every usable pass has been extracted from the event data, and we've extended all observations in the dataset with an extra intended receiver field which will be used as the de facto intended receiver for our feature extraction in the next chapter.

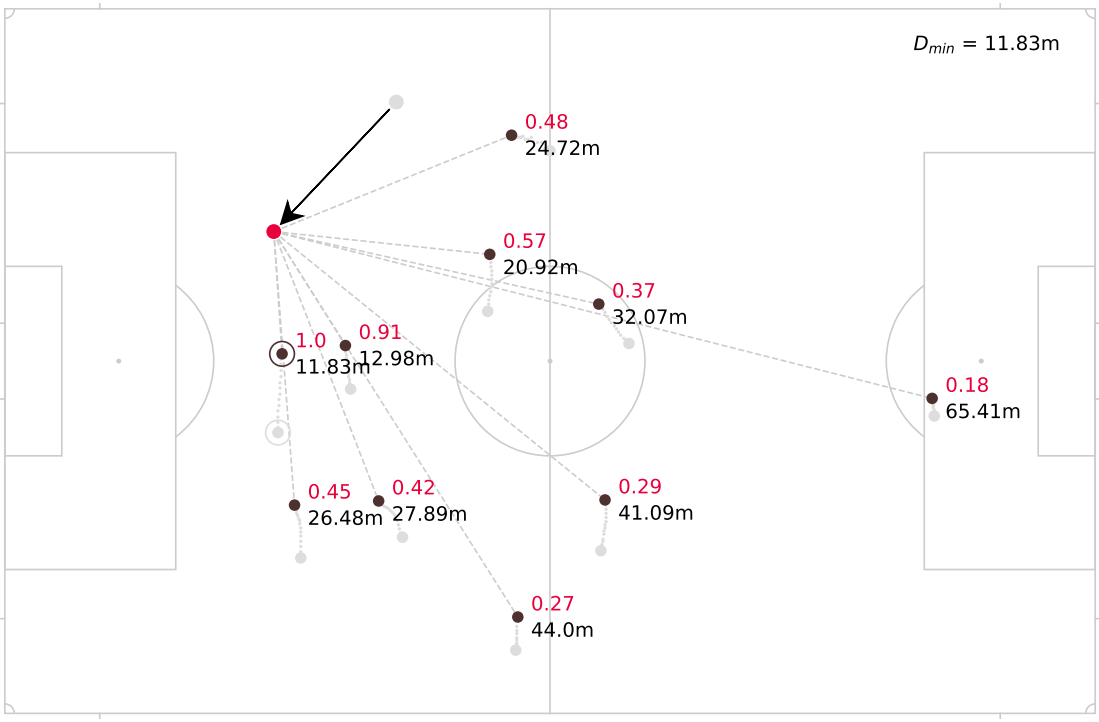


Figure 4.4: Distance component of the intended receiver formula. All team member positions at pass end time are shown. For each team member the distance to the actual pass receiver D_p (bottom) and the distance component score $\frac{D_{min}}{D_p}$ (top) are given

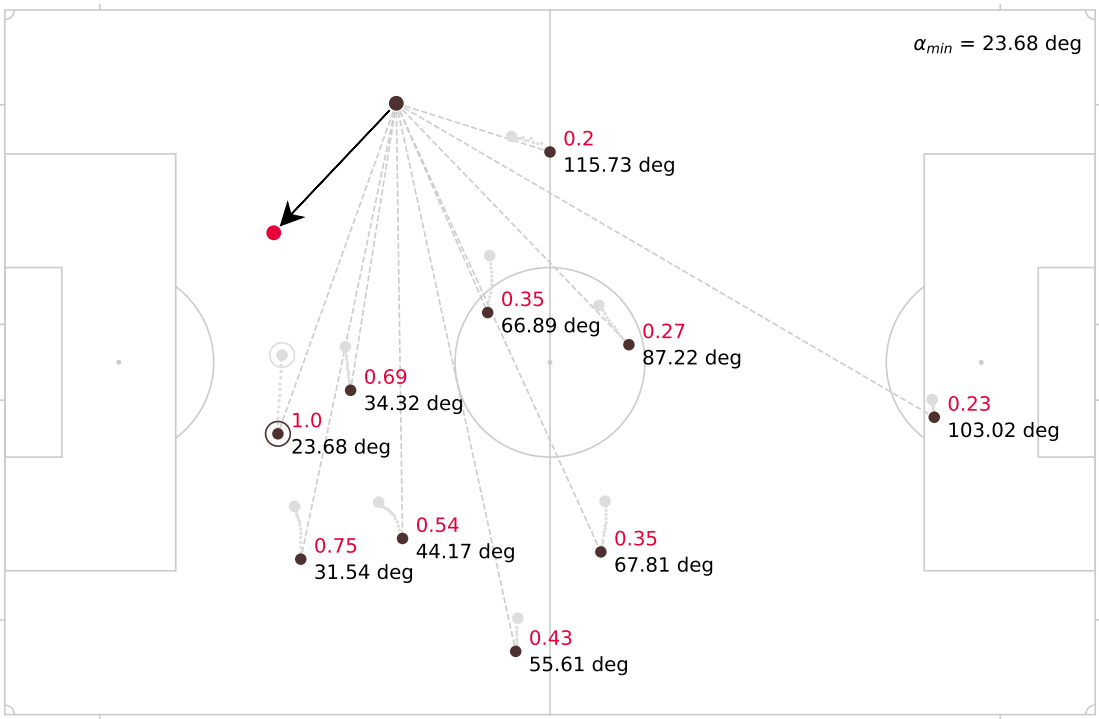


Figure 4.5: Angle component of the intended receiver formula. Team member positions at pass start time are shown. For each team member the angle between the hypothetical and actual pass vector α_p (bottom) and the angle component score $\frac{\alpha_{min}}{\alpha_p}$ (top) are given

Chapter 5

Feature engineering

In the previous chapter, we discussed the various Amisco tracking system datasets and in particular we described the steps undertaken to extract a pass dataset from the event data. For machine learning purposes, each pass needs to be converted to a series of numerical descriptors called *features* which describe the characteristics of the pass.

Since our goal is to create a predictive model for pass outcomes, we are going to focus on features that can be calculated at the starting moment of the pass. In this chapter, we will describe how we use the Amisco position data to calculate these new features for the passes extracted from the event data. We'll give a description of all the features implemented, spread over 4 major categories - *pass participants*, *pass characteristics*, *opponents' influence* and *contextual information*

5.1 Pass participants

The first major category of features describes the state of the pass participant themselves - the sender and intended receiver of the pass.

Player velocities The first features describe the velocities of the players involved in the pass. Specifically, we look at the velocities at the start of the pass. Velocity can be split up into 2 mathematical components: speed and direction. While speed describes the intensity of the velocity, direction describes where the velocity is directed towards within the coordinate system of our field. The speed of a player is calculated by using the distance that is covered between the previous and the current timestamps of the position data. The direction is calculated by constructing a vector from the previous timestamp's position to the current timestamp's position and using basic trigonometric functions to get the angle of that vector.

While it is possible to rely on one value (based on 2 datapoints) for speed and one value for direction, this approach has significant drawbacks such as a lack of robustness [46]. By using a window for the calculated speed and direction values and aggregating these values we can get a more robust approximation of the true value.

We then need to figure out how big our window should be. The authors of Ref. [48] explored an adaptive approach that changes the size of the time window based on the amount of variation around the timepoint of interest. For our use case, we instead decide to go for a simple, but robust, approach by using a fixed window with an uneven size which has the current timestamp at its center. We are only interested in an uneven

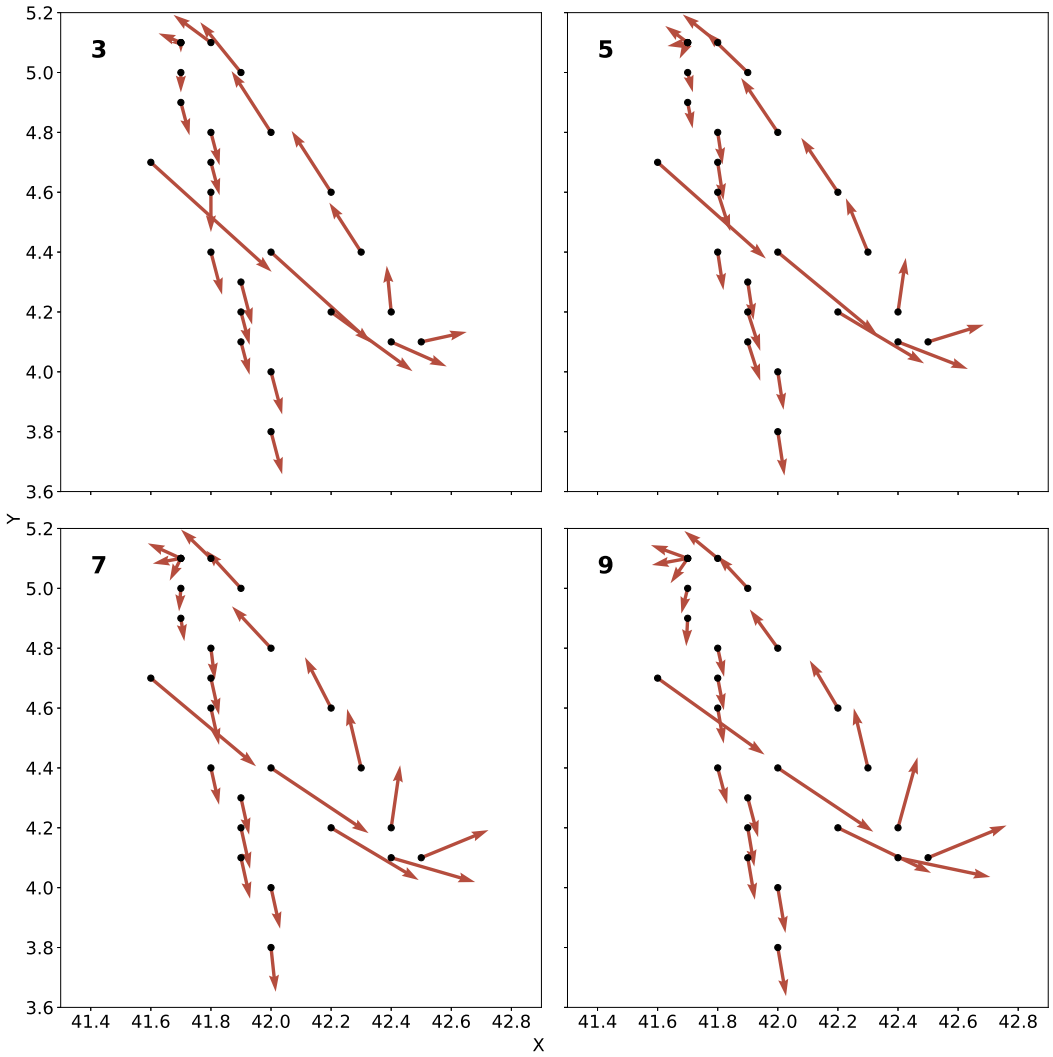


Figure 5.1: Velocity vectors for a player moving and making a sharp turn, calculated with window sizes of 3, 5, 7 and 9. The origin of each vector marks the player's field position at a certain moment while the attached arrow describes the direction and intensity of the player's velocity at that same moment (in m/s)

window size because we want the timestamp of interest to be at the center of the window to avoid bias in our velocity calculations.

We calculated player velocities based on several window sizes and the results of this procedure can be seen in figure 5.1. It can seen that increasing the window size beyond 5 does not affect the robustness of the velocity vectors nearly as much as increasing the window size from 3 to 5. Based on these results we chose a window size of 5 as the best tradeoff between calculation robustness and amount of data required. In order to calculate a final speed value and a final direction value, we calculate the speed and direction values for every consecutive pair of points in this window and then take the arithmetic mean of the speed values and the mean of circular quantities [28] of the direction values.

Finally, while an absolute direction value is useful, we think that such a value can have a radically different meaning depending on the player's team allegiance. As such, we convert the calculated player direction into a relative direction value that takes the

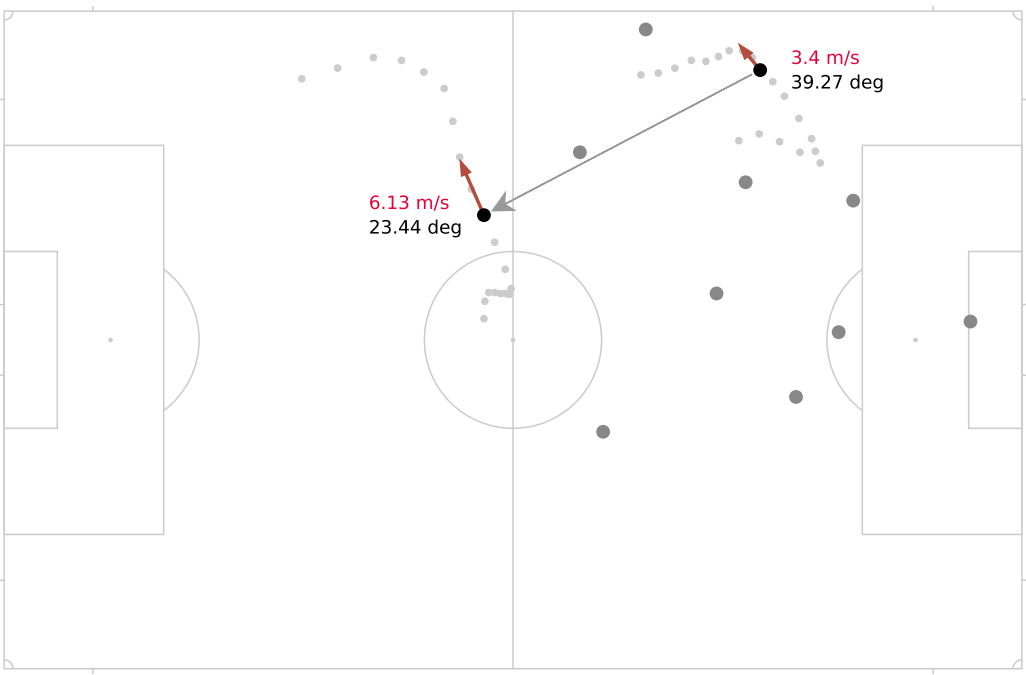


Figure 5.2: Example of player velocities calculated for the pass sender and receiver. The calculated speed and direction are shown and each player's path around the pass start moment is visualized as a series of light gray dots.

player's team's play direction into account - more positive values indicate the player is moving towards the opposing team's goal line. Moving directly towards the opposing team's goal line gets assigned a value of 90 while moving backwards towards the own goal line is assigned a value of -90. An example pass has been highlighted in figure 5.2 with the associated player velocity features.

Player positions In addition to player velocities, we are also interested in the actual position of the players at the time of the pass. Absolute coordinates give us symmetrical results, with similar patterns occuring at opposite sides of the field, due to the two teams playing in opposite directions. To account for this, we transform the absolute coordinates into relative coordinates using the player's team's direction of play. Essentially, we invert the sign on the X-coordinate if the player's team plays towards the opposite direction. Now a higher X-coordinate always means the player is positioned further away from his team's goal line, with said goal line being associated with a X-coordinate of -52.5 while the opposing team's goal line is associated with a X-coordinate of 52.5.

Additionally, for the Y-coordinate we found that unsuccessful passes tend to be more common when the pass sender is close to either side line and the intended receiver is close to the middle of the field, resulting in another symmetrical pattern. In order to make the Y-coordinate more easily interpretable by a classification model, we will take the absolute value of the coordinate. This results in a higher value being associated with the player being further away from the middle of the field. A Y-coordinate of 34.0 is now associated with both sidelines.

We've now given an overview of all the pass participant features. Appendix B.1 contains a table which describes all the mathematical features touched upon in this section, including their textual descriptions, format and identifiers.

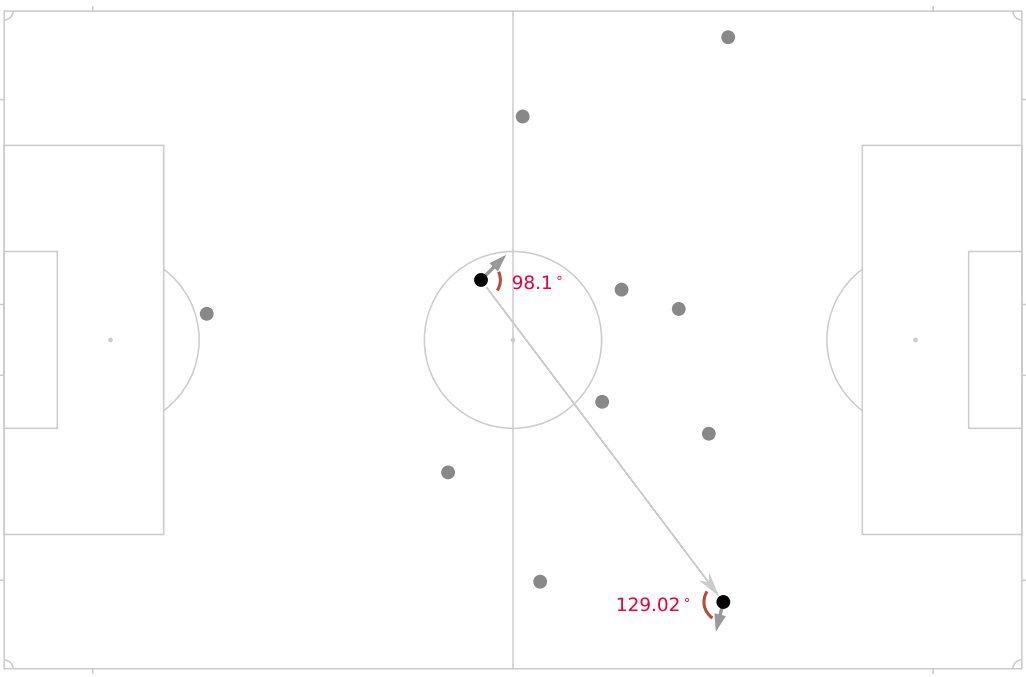


Figure 5.3: Example of player pass angles calculated for the pass sender and receiver.

5.2 Pass characteristics

The second major feature category contains descriptors of the pass vector and how it relates to the players taking part in the pass. For the remainder of this section, we will assume the pass vector is defined by the pass sender's position and the intended receiver's position at the start of the pass.

Pass trajectory A pass can be defined in terms of the spatial properties of its trajectory. We will focus on the distance and direction of the pass. Both of these values can be inferred based on the position of the pass sender and the intended receiver at the start of the pass. The ball will have to traverse the space between the two players and reach the intended receiver in order for it to be considered a successful pass.

The pass distance is considered to be equal to the length of the pass vector. Just like the direction component of the player velocities feature, the pass direction can be interpreted differently depending on the pass sender's team allegiance. For this reason we will again adjust this directional value by making it relative to the passing team's play direction, ranging from -90 (backwards) to 90 (forwards).

Pass angle The next features we add are related to the direction of the pass relative to the direction of the players. So far we've looked at the direction angle of each player's velocity, which tells us something about the game context, and the direction angle of the pass vector, which contains information about the type of pass attempted.

We consider the pass angle to be the smallest angle between the pass vector and the player velocity vector for the player in question. This feature is calculated for both the sender and intended receiver and an example calculation of these pass angle features has been illustrated in figure 5.3.

We've now gone through all the pass characteristic features and to summarize this section a table has been added in appendix B.2 which again lists all the discussed features and their formats.

5.3 Opponents' influence

So far we've looked at features focused on the players involved in the pass or the spatial properties of the pass. There's another very important variable that hasn't been addressed yet, which are the opponents. This is important for passes as they are a typical moment in which opponents might try to intercept and win over the ball. In this section we will describe our attempts at including information about the opponents and their influence on the pass.

Nearest opponent to player How far away the opponents are from the players involved in the pass is a basic feature that can already tell our model a lot about how much influence the opposing team can exert on a pass. We implement a feature which simply describes a pass participant's distance to the nearest opponent, describing the minimum distance any opponent has to cover to get to said player. We add this feature for both the pass sender and the pass' intended receiver.

On top of this, we also add a feature describing how fast the closest opponent is moving towards the player involved in the pass, as a proxy for the "preparedness" of said opponent. We calculate this value by taking the nearest opponent's velocity vector and projecting it in the direction of the pass participant it is closest to. The projected vector's length is then used as the projected speed towards the player. The calculation of these features has been plotted in figure 5.4. Notice how the distance of the nearest opponent essentially marks a "free zone" around the pass participant.

Nearest opponent to pass vector We just described features which look at the nearest opponent to both the sender and intended receiver of the pass. While this captures a lot of information in one value, describing the radius of an implicit "free" zone around each player, there's still another spatial component opponents can exert influence on - the pass trajectory itself. Instead of calculating the distance to a point, we now calculate the distance to the line segment representing the pass vector. Again, we consider the pass vector to be the vector between the pass sender and intended receiver's positions at the moment of pass start.

Intuitively we know that when the ball gets passed, it has a travel time before it arrives at the intended receiver. During this travel time opponents that are closer to the intended receiver have a bit of time to run into the pass vector to attempt to intercept. Since the previously described nearest opponent's distance metric does not capture this temporal aspect, we now also propose an alternative feature that does not have this shortcoming. This feature uses the angle between the pass vector and the vector from the pass start position to the opponent's position, instead of the opponent's distance to the pass vector. Again we use the smallest value found among all opponents. Both the distance and angle based features described in this subsection have been highlighted in figure 5.5.

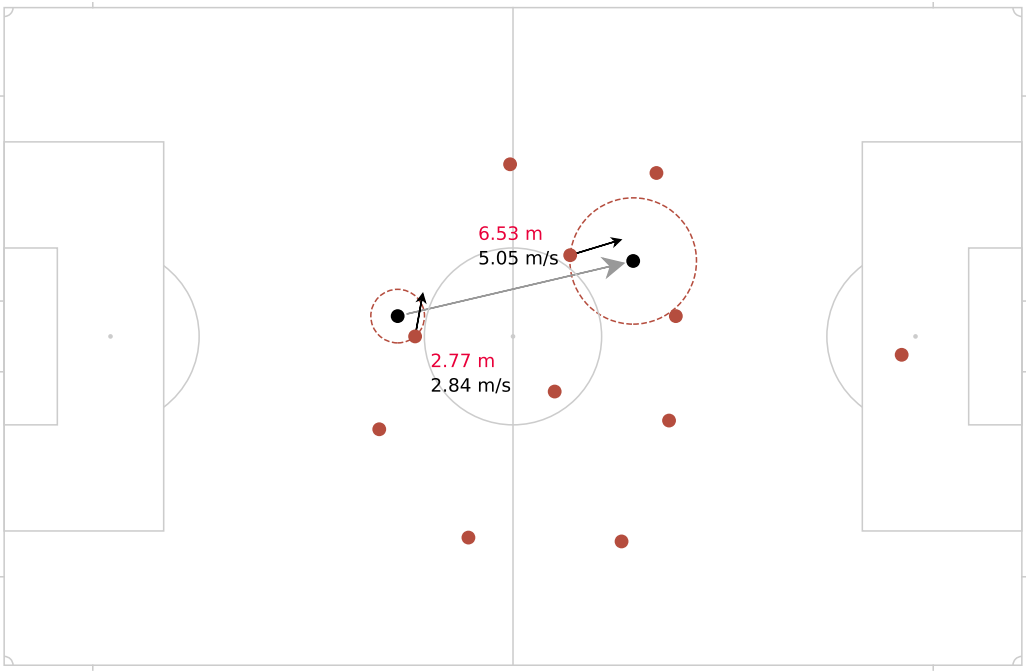


Figure 5.4: Example of the nearest opponents calculated for the pass sender and receiver. The nearest opponent's distance is visualized with a circle around each player, and the velocity vector of the nearest opponents are also shown as arrows. The calculated distance and projected speed towards the player in question are shown.

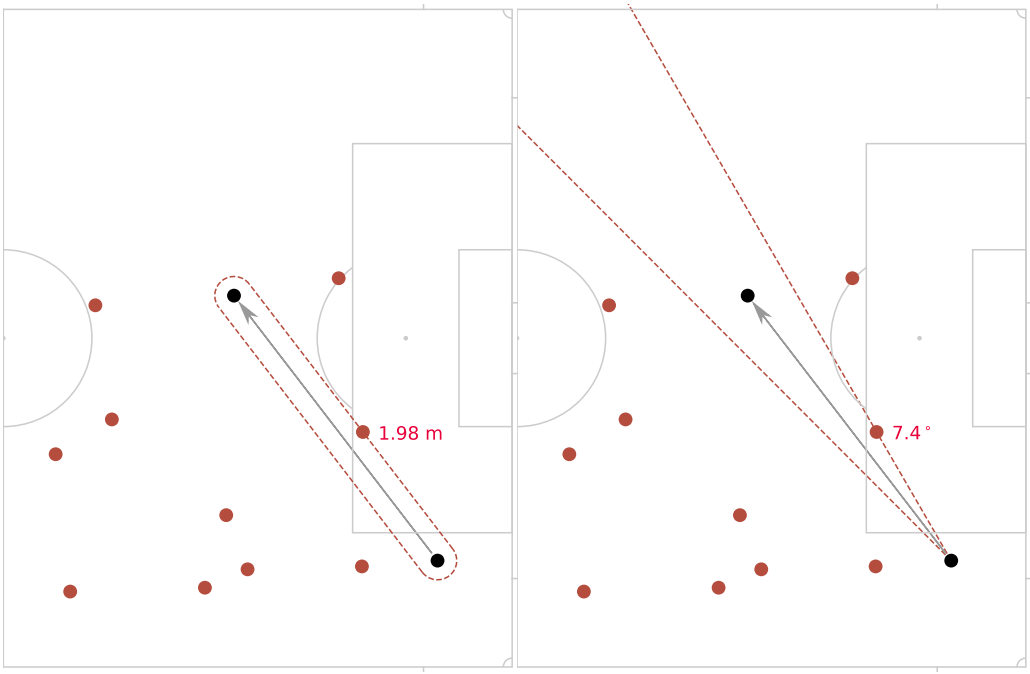


Figure 5.5: Example of the nearest opponent calculated for the pass vector. The opponent's distance and angle to the pass vector are shown.

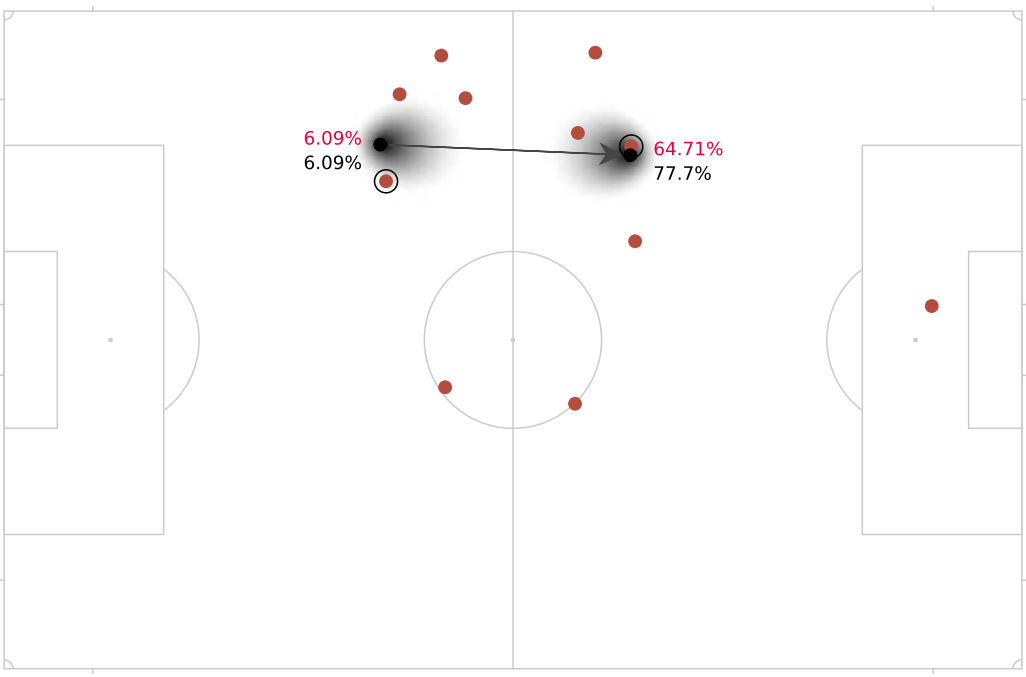


Figure 5.6: Example of opponent pressure calculated for the pass sender and receiver. Each opponent adds between 0% and 100%, depending on their location within the pressure zone. Both the pressure exerted by the most pressuring opponent (highlighted) and total pressure exerted by all opponents within the pressure zone are shown.

Pressure on player While the distance to the nearest opponent for both players involved in the pass gives us a measure of how "free" a player is, this metric fails to take any directional information into account. Moreover, the metric also fails to take information from more than one opponent into account. For example, if two opponents are equally close to the sender of the pass, it is considered similar to a situation where only one opponent is that close to the sender. This leads us to search for a numerical representation of "pressure" that can capture this detail.

A general definition of pressure is discussed in Ref. [2] as a possible way to analyze player behaviour and team tactics. The authors use their pressure metric to visualize which player tends to pressure which player of the other team, as well as the ball, during specific sequences in a football match. This enables them to explore a team's tactics in a more in-depth way. Using their metric, a pressure value can be assigned to each opponent at any moment which tells us something about how effectively said opponent can disrupt opportunities, such as passes or shots on the goal. It takes into account directional information about where exactly the opponent is relative to a "threat direction". This threat can be a pass, which is very useful for our work.

Applying this metric to our problem gives us an egg shaped pressure zone around the player involved in the pass, with the long side following the pass vector away from the player. Opponents that are behind the player involved in the pass have a much smaller radius in which they can pressure the player. We will calculate two variants of this feature for each of the two participants - the most pressure exerted by a single opponent and the total pressure exerted by the entire opposing team. In figure 5.6 we've highlighted what the output of such a calculation looks like.

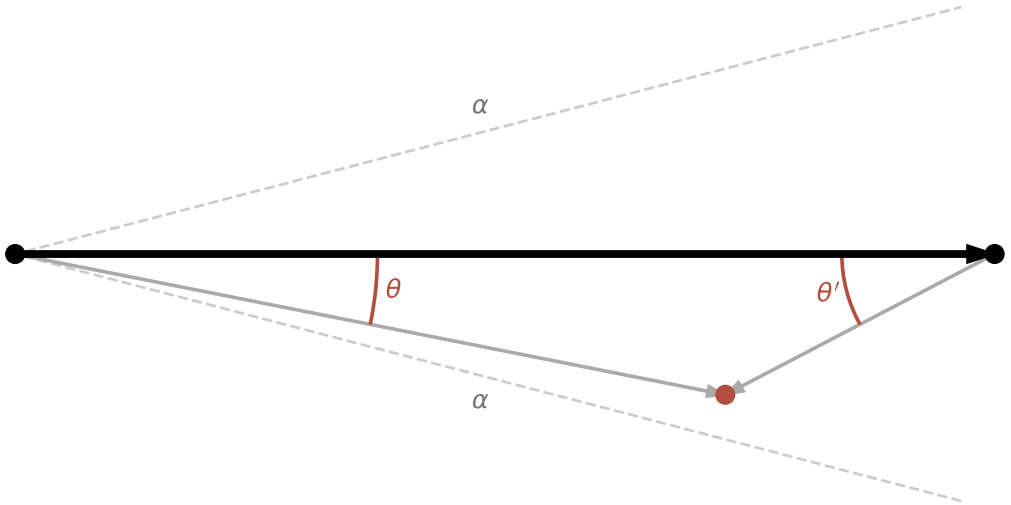


Figure 5.7: Illustration of the parameters used in the pass vector pressure calculation for a specific opponent. The sender, receiver and pass vector are highlighted in black while the opponent is highlighted in red.

Pressure on pass vector We already looked at a metric describing how far away the closest opponent is from the pass vector. This metric can be seen as a proxy for how "free" of opponents' pressure the pass vector is. Just like with the nearest opponent metric for individual pass participants, the main downside of using this metric is that it only describes the closest opponent and any other opponents are completely ignored.

Similarly to how we calculated a pressure feature for the nearest opponent to each pass participants, we'll now define a pass vector pressure function that scales up in value as more opponents are within the pass' pressure zone. We want to incorporate the previously mentioned temporal aspect of pass vectors, so we define a conal function that takes the angle between the pass vector and the vector between the pass start position and opponent's position into account. Due to a lack of existing literature on this, we define our own function as follows:

$$p = \begin{cases} \left(1 - \frac{\theta}{\alpha}\right)^q \cdot 100 & \text{if } \theta' < \frac{1}{2}\pi \\ 0 & \text{otherwise} \end{cases}$$

where θ is the angle between the pass vector and the sender-opponent vector, θ' is the angle between the pass vector and the receiver-opponent vector and $\alpha = 15°$ is the angle limit at which the pressure exerted by the opponent reaches zero. These parameters have been highlighted in figure 5.7. $q = 1.75$ is the constant speed of angle decay and determines how quickly pressure drops off as the angle value θ' increases. For this parameter we use the same value as was used as decay value for the player pressure definition in Ref. [2]. We acknowledge that further optimization may be done here, but leave this open for future work.

The output of this calculation has been visualized for an example pass in figure 5.8. Notice the conal pressure zone and the pressure-free area behind the receiver.

We've now gone through all of the features describing the influence of the opposing team's players. A comprehensive overview of all these mathematical features can again be found in appendix B.3 in a tabular format, including their descriptions and formats.

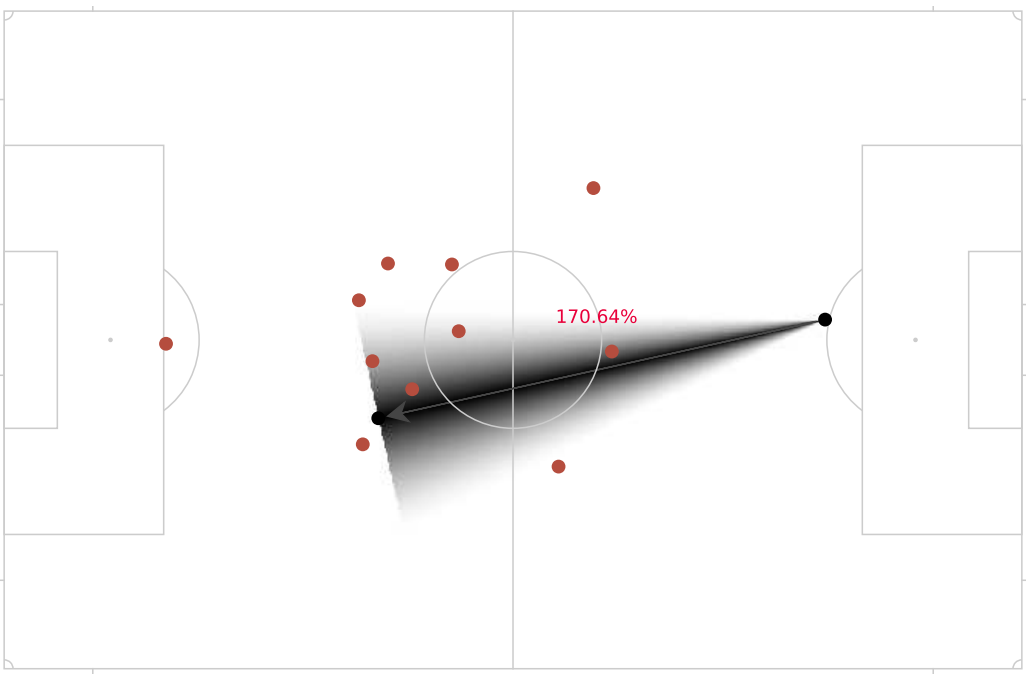


Figure 5.8: Example of opponent pressure calculated for the pass vector. Each opponent adds between 0% and 100% to the calculated value, depending on how close to the pass vector they are within the black pressure zone.

5.4 Contextual information

We've now looked at various features describing the players involved in the pass, the pass vector itself and the players of the opposing team. The final category of features is that of contextual information that does not directly describe any of the spatial aspects of the pass, but instead describes how the situation fits within the context of the match as a whole. Features that describe how, and by whom, the pass was performed are also included in this category. In this last section, we'll describe the implemented features that fall under this category.

Ball possession time Ball possession is an important aspect of game context and we include information about it by implementing ball possession time as a feature. A similar feature for team ball possession has also been used by the authors of Ref. [41] and we will build on this by adding features for both player possession time and team possession time. These values can be derived from the Amisco event data with relative ease, by looking back to the last event in the data that was associated with a different player, or team, before the current moment. This event's timestamp can then be subtracted from the pass event's timestamp, giving us the time the current player or team has had possession.

Pass technique In the Amisco event data a textual description is listed which describes exactly how a pass was performed, ranging from left/right foot to headers. We directly copy this textual description, adding the pass technique as a categorical feature.

Passing team In addition to pass technique, we add the identity of the team as a categorical feature. We have six matches worth of data available and a specific team took part in each of them while the other five teams are that team's opponents in each of the six matches, with one opposing team playing against them twice. Essentially this means that we have six matches worth of data for one team, two matches worth of data for another team, and one match worth of data for the four remaining teams. This means one team will be significantly overrepresented in our dataset.

Player role The final feature we will look at is that of player role. The players in a team have different roles such as defenders and forwards, with these roles having distinct subtypes as well. Each pass in our dataset has a pass sender an intended receiver and both of these players have a role within their team. We would like to add a feature for each of these two players describing what specific role they hold within their team. Deriving contextual information like this from the position data is a challenging problem that is worth its own research project. For example, the players of a team can be mapped to 11 unique roles using a minimum entropy data partitioning method, using spatiotemporal position data [9].

In our work we have not implemented automatic position data player role detection. Since we only have six matches worth of data available, we opted to instead augment the player metadata (described in section 4.3) with explicit role definitions, added by hand based on each player's identity and their roles described in the official technical report released by the UEFA. The technical report we have access to assigns one of four possible roles to each player in each team: Goalkeeper, Defender, Midfield and Forward. We assign one of the four roles to each player present in the match and then assign the role of the pass sender and the role of the intended receiver as two distinct categorical features.

We've now finished describing all of the features containing contextual information. Before ending this chapter, we refer the reader to a full overview of the calculated features and their formats in a table in appendix B.4.

5.5 Dataset Finalization

After calculating all of the features described in this chapter, we still need to make sure the final dataset is entirely usable by our machine learning algorithm of choice. Most algorithms can't handle missing input values, so passes with missing values need to be cleaned up. Only 51 of the 4158 observations contain missing values, so we remove the incomplete observations.

Additionally, in figure 4.3 it was shown that we have a mildly unbalanced dataset where 68% of all observations are of the positive class. To compensate for this, we use random upsampling to add more observations of the unsuccessful pass outcome class to the dataset. This method will be used during the training procedure to avoid bias towards the majority class in our balanced model.

Chapter 6

Results

In the previous chapters we talked about the entire machine learning pipeline we built to do data preparation, feature extraction and predictive modelling. This pipeline takes the Amisco data as input and gives us a trained predictive pass model and a detailed model generalization performance assessment as output.

In this chapter, we will evaluate the distributions of several promising features that were generated and assess the performance of the trained models according to conventional classification metrics. Additionally, the importances of the individual features will be evaluated using the best model and special attention will be paid to the probablistic output of said model.

6.1 Generated features

29 features have been generated, each describing some numerical or categorical property of a pass. We now look at the distributions these feature generate over the entire pass dataset and whether there is a significant difference between the distributions for successful and unsuccessful passes. To support this we perform the statistical tests described in a previous chapter. In doing so we calculate a p-value for every feature.

We found that only three out of 29 features have no statistically significant differences between the successful and unsuccessful pass distributions, with p-values over 0.05, while all other features have siginificantly lower p-values. The six features with the lowest p-values have been highlighted in figure 6.1, with both types of passes having their own density plot. By using a density plot for each individual type of pass we compensate for the lower number of unsuccessful passes and make both classes more easily comparable. It can be seen that all six features have different distributions for the two different classes to varying degrees.

For example, the nearest opponent's distance to receiver feature (opp_distance_r) has a right skewed distribution for both pass types, while the distribution of positive outcomes has a much longer tail and a much smaller peak, implying that passes with a higher distance tend to be much more closely associated with positive outcomes.

The features related to pressure have an extremely skewed distribution due to the presence of zero-values. Any pass with no opponent in the pressure zone is assigned a value of zero for this feature, and there are a lot of those types of passes. For both highlighted pressure features we can observe that the zero-valued passes and passes with a value close to zero tend to have positive outcomes, while the passes with higher values for these features tend to have negative outcomes.

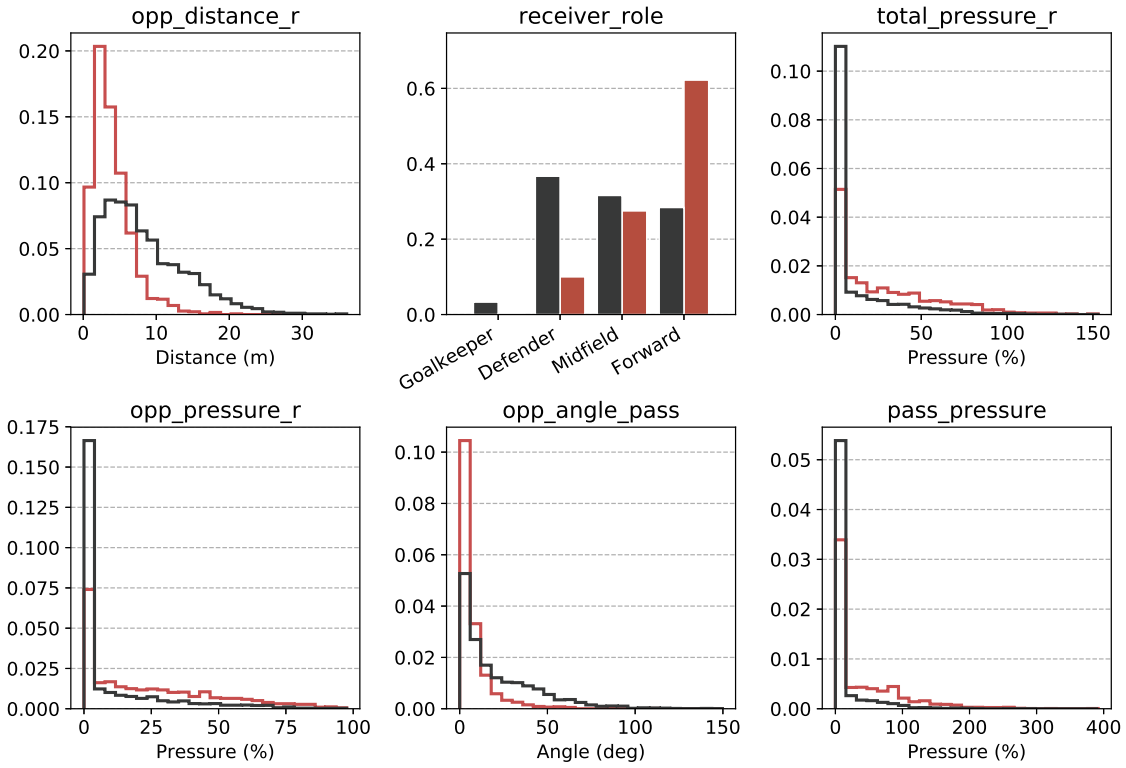


Figure 6.1: Density plots of both successful and unsuccessful passes for the six features with the lowest p-values. Black lines/bars correspond with successful passes while red lines/bars correspond with unsuccessful passes

What we can observe from all of these distributions is that there is a significant difference between the two different types of passes and how they interact with the features in question.

6.2 Model performance

Several classification models have been trained on the complete dataset, ranging from logistic regression to ensemble tree methods. The full reports of the model training procedures for these models have been added in appendix C. These reports include a detailed description of the nested cross-validation process and the hyperparameters that were selected for every outer cross validation iteration. Additionally, a side-by-side comparison of the accuracy, precision and recall of all models is available in table 6.1.

All models beat the 68% accuracy baseline by a comfortable margin. Logistic regression, AdaBoost and support vector machines all seem to have similar performance, with an accuracy around 74%, while random forest performs slightly better with an accuracy of 75.7%. On the other hand, k-nearest neighbors performs noticeably worse with an accuracy of 71.2%.

More specifically, it can be seen that the three models with similar accuracy also all have similar precision and recall values, which implies that they behave in a similar way. Random forest instead seems to have somewhat higher recall on the positive passes **p** and significantly lower recall on the negative passes **n**. In addition to that, the precision on **p** went down noticeably, dropping from 0.88 (for logistic regression) to 0.82, while the precision on **n** went up considerably.

Model	Accuracy	Class	Precision	Recall
Baseline	68.08%	**p**	0.68	1.00
		n	-	0.00
Logistic regression	74.38%	**p**	0.88	0.73
		n	0.57	0.78
AdaBoost	74.12%	**p**	0.86	0.74
		n	0.57	0.74
Random Forest	75.68%	**p**	0.82	0.82
		n	0.62	0.62
Support Vector Machine	74.07%	**p**	0.86	0.74
		n	0.57	0.75
K-Nearest Neighbors	71.15%	**p**	0.85	0.70
		n	0.54	0.73

Table 6.1: Performance of our trained models and the baseline. **p** and **n** represent the subsets of successful and unsuccessful passes, respectively

In order to compare random forest to the other models in a more detailed way, we will compare its confusion matrix to the confusion matrix of logistic regression in figure 6.2 where successful passes **p** and unsuccessful passes **n** are compared to their predictions. These confusion matrices are constructed by collecting the results of the test folds in the outer cross-validation loop in our model assessment procedure.

Logistic Regression

	$\hat{\mathbf{p}}$	$\hat{\mathbf{n}}$	**total**
p	2029	763	2792
n	289	1026	1315
total	2318	1789	4107

Random Forest

	$\hat{\mathbf{p}}$	$\hat{\mathbf{n}}$	**total**
p	2293	499	2792
n	500	815	1315
total	2793	1314	4107

Table 6.2: Confusion matrices of our logistic regression and random forest models. **p** and **n** represent the ground truth while $\hat{\mathbf{p}}$ and $\hat{\mathbf{n}}$ represent the predictions for successful and unsuccessful passes, respectively

A difference that should immediately be obvious is that the random forest model indeed classifies far more passes as being positive compared to the balanced model - 2793 versus 2318. This highlights that the random forest model picks up on the successful passes being the majority, despite the dataset balancing that has been done beforehand. As can be seen in table 6.1, the tradeoff here is a higher accuracy for a worse precision on the majoity class. The opposite is then true for the negative passes **n** - the much smaller number of passes it still predicts as negative tend to be more likely to actually be negative passes.

While the random forest performed slightly better in terms of accuracy, we will use logistic regression for the remaining results in this chapter, due to its transparency and probabilistic nature.

6.3 Feature importances

In the previous section, we looked at the predictive potential our model has for classifying the outcome of passes. While this is one of the primary purposes of classification models, there are other uses such a model has. In this section, we'll look at model interpretability and more specifically the interpretation of feature importances. By looking at the importance of specific features in a model, we can derive out what the most important factors influencing the outcome of a pass are according to our model.

6.3.1 Feature weights

Logistic regression models assign a weight to each input feature indicating how much influence it has on the final prediction, per unit of said feature. Since we've normalized all of our input features, we can directly interpret the weights of the features as the relative importance the model places on each weight. The six highest absolute feature weights of our model are listed in table 6.3.

Feature name	Feature ID	Weight
Nearest opponent's distance to receiver	opp_distance_r	0.957045
Nearest opponent's distance to pass	opp_distance_pass	0.587692
Pass direction	pass_direction	-0.533240
Pass distance	pass_distance	-0.492576
Total pressure on passer	total_pressure_s	-0.392761
Most pressure on passer	opp_pressure_s	0.329652

Table 6.3: Highest six feature weights of our final balanced logistic regression model trained on normalized data

It seems like the *Nearest opponent's distance to receiver* feature (opp_distance_r) is the strongest feature by far. However, before we can interpret these weights as the relative influence of individual features on the probability of pass success, we have to assess the degree of collinearity between the features.

6.3.2 Feature collinearity

In table 6.4 the variance inflation factors (VIFs) for the six features with the highest VIFs in our dataset are shown. As previously mentioned, a general rule of thumb is that VIFs with a value over 5 are considered to be high and indicate that a feature is highly collinear with the other features in the data.

Feature name	Feature ID	VIF
Receiver X	x_r	28.008024
Sender X	x_s	26.339375
Total pressure on receiver	total_pressure_r	16.105080
Most pressure on receiver	opp_pressure_r	15.512434
Total pressure on passer	total_pressure_s	11.572538
Most pressure on passer	opp_pressure_s	11.352844

Table 6.4: The six continuous features with the highest Variance Inflation Factors. Higher values imply a greater degree of collinearity with the other features in the dataset

As can be seen, some of the features with significant weights have very high VIFs. The weights assigned to them are highly related to the weights assigned to features collinear with them, and this prevents us from directly interpreting the weights as the relative importance of those factors in the outcome of a pass.

6.3.3 Bivariate models

Due to the significant feature collinearity, we've instead trained individual bivariate models that attempt to predict pass outcome based on single features. Such models will give us a much better idea of how strong individual features are, as we avoid the overlap in variance explanation of different features by analyzing each feature's performance individually. In order to assess each feature fairly, we've trained a logistic regression model for each individual feature using the nested cross-validation procedure previously discussed. For categorical features that were transformed into dummy variables using one hot encoding, we include all dummy variables in one model. The six best performing models, in terms of accuracy, are shown in table 6.5.

		p		**n**	
Predictor ID	Accuracy	Precision	Recall	Precision	Recall
pass_pressure	0.724	0.765	0.859	0.594	0.439
Pass technique	0.695	0.716	0.912	0.555	0.234
total_pressure_r	0.693	0.772	0.780	0.522	0.510
opp_pressure_r	0.693	0.775	0.774	0.521	0.522
Receiver role	0.686	0.801	0.716	0.508	0.622
opp_distance_r	0.672	0.864	0.614	0.492	0.795

Table 6.5: Performance metrics of the six single-predictor logistic regression models with the highest accuracy. **p** and **n** represent the subsets of successful and unsuccessful passes, respectively. Feature names listed in italic are categorical features of which all one hot encoded sub-features have been used for training the model

We can immediately see that the results are very different when compared to the feature weights of the main logistic regression model shown previously. While the *Nearest opponent's distance to receiver* feature (opp_distance_r) had the heighest weight by far, in this bivariate modelling step the model trained on it does not produce the best performing model at all when measured in accuracy (though it does produce high precision on predictions of the positive class). Instead, the *Pass vector pressure* feature (pass_pressure) has the highest accuracy by far. Interestingly, this feature had a relatively minor weight in the final model trained on all features together.

6.3.4 Trivariate models

We now train models with two predictors to see how the performance of the models improves as we add more features. We once again train the models using the familiar nested cross validation procedure and we train one model for every possible pair of features. This is a pretty time consuming procedure that gives us hundreds of trained models. In table 6.6 we've listed the three best trivariate models (again based on accuracy) with the pair of features involved in each.

			p		**n**	
1st predictor ID	2nd predictor ID	Accuracy	Prec.	Recall	Prec.	Recall
pass_pressure	team_possession_t	0.731	0.769	0.864	0.609	0.449
pass_pressure	pass_distance	0.731	0.770	0.861	0.607	0.455
Pass technique	pass_pressure	0.729	0.791	0.818	0.583	0.541

Table 6.6: Performance metrics of the 3 logistic regression models with 2 predictors with the highest accuracy

			p		**n**	
1st predictor ID	2nd predictor ID	Accuracy	Prec.	Recall	Prec.	Recall
opp_distance_r	pass_distance	0.707	0.859	0.681	0.530	0.762
pass_pressure	opp_distance_r	0.707	0.858	0.683	0.530	0.760
opp_distance_pass	pass_direction	0.678	0.856	0.632	0.498	0.775

Table 6.7: Performance metrics of the 3 logistic regression models with 2 predictors with the highest precision

Closely matching our results from the bivariate analysis, the *Pass vector pressure* feature (pass_pressure) is a part of all the best performing features. In fact, the 20 pairs with the highest accuracy are all pairs including this feature.

In table 6.7 the three models with the highest precision have been highlighted. We can see that models in which the *Nearest opponent's distance to receiver* feature (opp_distance_r) is involved, tend to perform very well. These models have significantly higher precision for the positive class of successful passes. For example, the model of the pass vector pressure feature and the *Nearest opponent's distance to receiver* feature (opp_distance_r) has a significantly higher precision than the other models involving the pass pressure feature. The model with the highest precision includes the *Nearest opponent's distance to the receiver* feature (opp_distance_r) and the *Pass distance* feature (pass_distance).

6.4 Probabilistic model output

Previously, we already looked at how good the predictive power of our model is for the purpose of classifying passes based on pass successfulness in a binary fashion. While this classification is important and one of the primary use cases for logistic regression, the fact that this algorithm is naturally probabilistic has other benefits. More specifically, we can use the model to output prediction probabilities, instead of classifications. Such probabilities can be used as a measure of confidence.

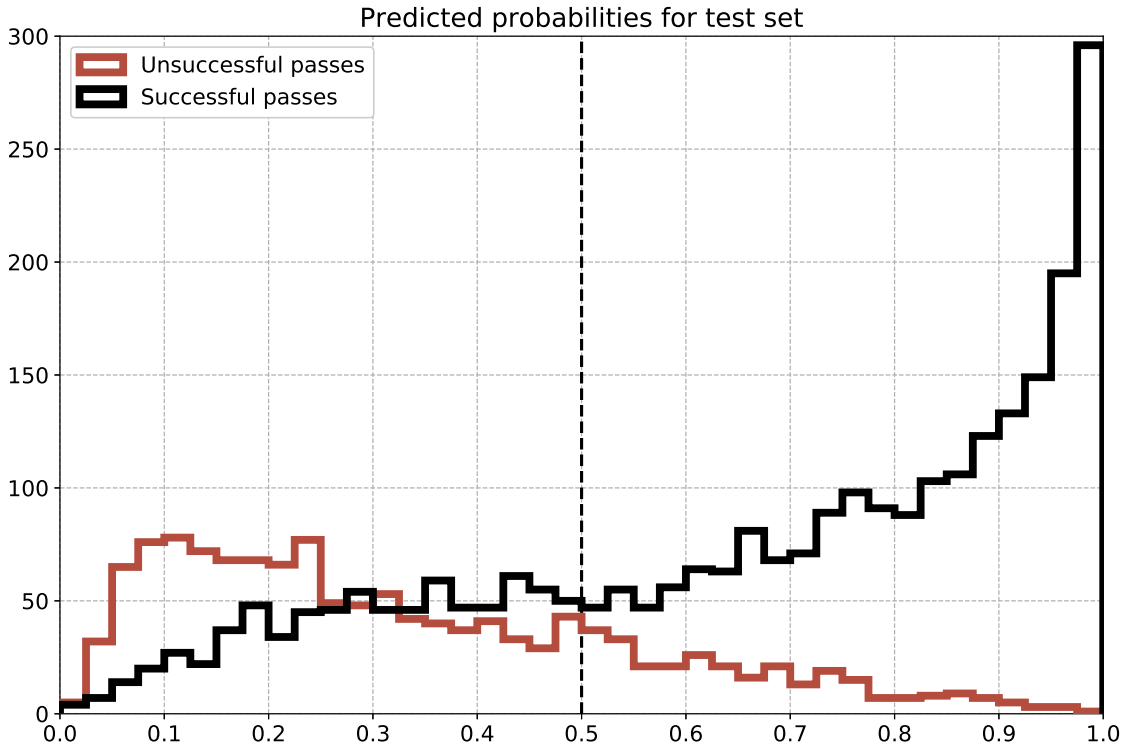


Figure 6.2: Histogram of predicted probabilities for all passes in our dataset during the outer cross validation loop, split into successful passes and unsuccessful passes. The 0.5 probability threshold, used for classification, is also highlighted

In order to get a sense of how useful the probabilities returned by our model really are, figure 6.2 plots the 2 histograms showing the distribution of probabilities for both successful and unsuccessful passes in our dataset. This distribution includes all passes in the pass dataset and describes the probability of pass successfulness associated with each pass during the outer cross validation loop of our nested cross validation procedure. We can see that the successful passes have a very clear curve that peaks at p=1.0, while there are very few successful passes close to p=0.0. On the other hand, the unsuccessful passes have a significantly different curve that does not actually peak at p=0.0 - instead it has a small peak at p=0.1 and then drops down again.

Chapter 7

Discussion

In this chapter, we will interpret and discuss the results shown previously. Particular attention will be paid to the importance of the individual features and the model's performance and value. Since our logistic regression performed reasonably well compared to the other algorithms, we'll also discuss the probabilistic output of this model.

7.1 Model performance

In section 6.2, we went over the performance of all 5 models that we trained using multiple machine learning algorithms. The performance metrics were compared in table 6.1 and the confusion matrices of the most accurate model, random forest, and logistic regression were compared in table 6.2. We found that the models performed very differently when it comes to metrics such as accuracy and precision, though both have an accuracy that is significantly better than the naive baseline model. For a classification model to find use in real world scenarios, the costs and benefits of the different types of incorrect and correct classifications need to be carefully considered. The cost of unexpectedly failing a pass in a high stakes football match can be incredibly costly. As such, we value the precision of the positive class predictions much more than any other metric in the confusion matrix and we believe the logistic regression model will provide more value to football players, coaches and sports analysts even though the random forest model has slightly higher accuracy.

There are several limitations related to model performance in our current work. We tried a number of machine learning algorithms, but some of the more powerful options available, such as neural networks, were not explored in-depth and may be a better choice for maximizing classification performance. These models have significantly more parameters to be optimized and can take a lot of time to train and assess, depending on said parameters.

There is also an upper limit to how well any model can classify pass successfulness, given the technical limitations of current state of the art sports position data. The position data itself only contains 2 dimensional coordinates describing the center of mass of every participant in the match. The finer details that make a decisive difference for the success of a pass, such as the orientation and the direction the player is looking towards are all important factors that can affect how effectively a player is able to send or receive the ball for a pass. The availability of such information is likely going to be an important factor in further improving models for predicting the outcome of sports match events.

For possible future work, deep learning could provide us with models that are able to capture the most complex relationships detectable in football position data. However, properly utilizing deep learning takes a lot longer and due to time constraints we could not commit to fully exploring these alternative models. Additionally, the amount of data that we have available is simply too little for deep learning to truly be able to detect very complex (and robust) patterns in the data. We believe that neural networks have a lot of potential and exploring them would be a worthwhile endeavor, if enough data is available.

7.2 Feature importances

In this section, we'll discuss how much the individual features we calculated contribute to the predictive power of the models, in order to evaluate which features are the most meaningful for pass prediction. We have not been able to find any prior detailed work on this kind of feature analysis for pass prediction, so we see this as a new contribution to the literature on this subject.

The highest absolute feature weights for our final model were shown in table 6.3. From this we can observe that the distance opponents are from the intended receiver (opp_distance_r) and pass vector (opp_distance_pass) are the most dominant features, closely followed by the distance (pass_distance) and direction (pass_direction) of the pass vector itself. All of the modelled relationships make intuitive sense, with greater distances to the nearest opponents correlating with better pass outcomes and longer passes that are more forward correlating with worse pass outcomes.

Table 6.5 highlights the performance of the best six bivariate models trained using one feature as its single predictor. This gives us an accurate picture of how individual features perform on their own. The *Pass vector pressure* feature (pass_pressure) has the highest accuracy. In fact, half of the best six models all use some kind of pressure-based feature as their single predictor. The *Nearest opponent's distance to pass vector* feature (opp_distance_r) doesn't perform as well as the pressure-based features, but it does have noticeably higher precision on successful pass outcome predictions.

To compare how well features synergize, table 6.6 highlights the performance of the best three trivariate models (in terms of accuracy). The *Pass vector pressure* feature (pass_pressure) is a part of all the best performing pairs of predictors. In table 6.7, it can be seen that combining this feature with the *Nearest opponent's opponent's distance to receiver* feature (opp_distance_r) gives us a slight drop in accuracy, but a significant increase in precision on positive passes. The joint distribution of this feature pair has been highlighted in the left plot in figure 7.1. Table 6.7 also reveals that the pair of features giving us the highest precision on positive passes was the pair formed by the *Nearest opponent's opponent's distance to receiver* feature and the *Pass distance* feature, the joint distribution of which has been highlighted in the right plot in figure 7.1. We can see that that higher proportions of successfulness tend to be associated with higher distances to the nearest opponent for the receiver and lower pass distances. This makes intuitive sense - if the pass distance is greater the ball has to travel for longer and the opponents have more time to intercept close to the intended receiver of the pass. Once again both models with the nearest opponent's distance feature exhibit high precision on successful pass outcome predictions.

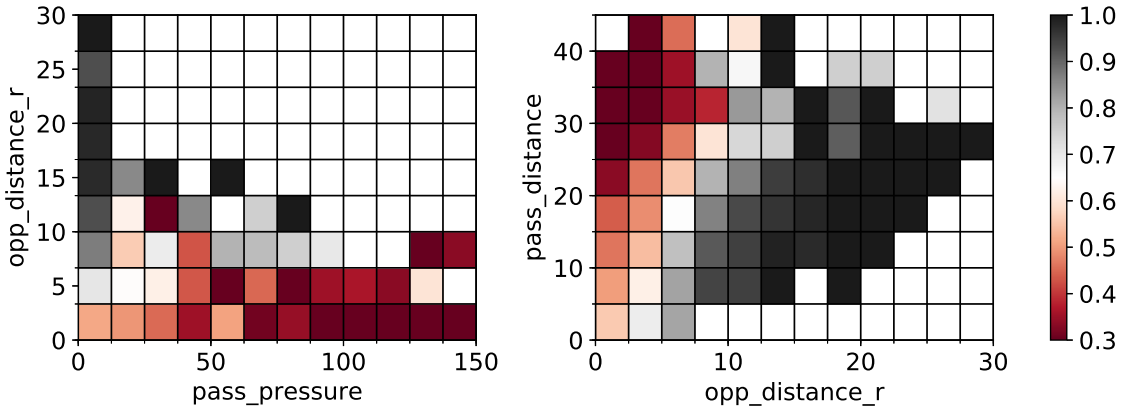


Figure 7.1: Pass successfulness proportion heatmap for several feature pairs including the *Nearest opponent's distance to pass vector* feature (opp_distance_r), based on the entire pass dataset

We noticed that the *Nearest opponent's distance to receiver* feature (opp_distance_r) was the best feature in the main model, but was not the strongest feature in the bivariate models at all. The feature has an unusually high precision on passes with a positive outcome, while still having respectable accuracy. The distribution of this feature has been highlighted in the top left plot of figure 6.1. We notice a right-skewed distribution for both types of passes, with the successful pass distribution having a smaller peak but a much longer tail. If the opponents are far enough away from the receiver it's unlikely a pass will fail.

The *Pass vector pressure* feature (pass_pressure) is the strongest feature among the bivariate and trivariate models, while this feature had an insignificant weight in the final model trained on all features together. The distribution of this feature can be seen in the bottom right plot of figure 6.1. We once again notice a right-skewed distribution, with the unsuccessful pass distribution having a longer tail this time. We also notice that many passes have a pressure value of 0 and these passes have a significantly higher proportion of successfulness. Pressure based features are inversely related to opponent-distance based features, with higher values correlating with worse pass outcomes.

Finally, all models seem to favor features that describe the influence of opponents and the characteristics of the pass vector. The metrics that are favored seem to vary depending on the scope of the model, which tells us there is significant overlap in the variance explained by individual features. For example, a bivariate model trained on only the *Pass vector pressure* feature has an accuracy of 72.4% which is not very far behind the 74.38% accuracy of the final model trained on all features. With many more features added one would expect a much bigger gain.

Several limitations are present in our current collection of features. Table 6.4 revealed that the variance inflation factors of our features are very high. This tells us some of the features in our pass dataset are highly collinear, which in turn means we cannot interpret our model weights as indicators of feature importance. The model weights of the final model are less informative because of this and several of the correlated features are most likely completely redundant. Another limitation is the lack of feature parameter optimization. Several parameter-based features were implemented in this work and for all of them we simply used default parameters defined in relevant literature. The pressure-based functions are some of the most powerful features we have available and all of them

have parameters to configure the limit of the pressure zone as well as the zone's decay rate. The lack of this feature parameter optimization is a considerable weakness.

One of the next steps would be to do proper feature selection and find the right balance between predictive power and simplicity of the model. This could help mitigate the significant collinearity present in our feature dataset. As part of this work, we added L1-regularization to our model, but unfortunately this did not remove a lot of features. We have not tried more dedicated feature selection procedures, such as Recursive Feature Elimination [24], which could significantly improve the simplicity of our model while not sacrificing much accuracy. Finally, optimizing the feature parameters in some systematic way has potential. This optimization could be incorporated into the hyperparameter optimization loop of our nested cross-validation procedure, but this would require a significant rework of our machine learning pipeline and the intertwining of the feature extraction and modelling steps, so we leave this open for future research as well.

7.3 Probabilistic model output

Logistic regression gives us the opportunity to evaluate the underlying probabilities used for the classification. We can use these probabilities to compare passes and get a better understanding of the pass outcome classifications. In figure 6.2, we highlighted the probability distributions produced by our model for all unsuccessful and successful passes in the dataset. It was noticed that the probability curve for unsuccessful passes has an unexpected lack of observations close to p=0.0.

There appears to be an innate decision bias in the pass dataset itself. Football players tend to avoid passes that are very unlikely to succeed, since maintaining ball possession is a crucial part of football. This bias does not go both ways, as a player has no reason to avoid a very safe pass unless a better alternative is available. This innate decision bias means that our pass dataset cannot possibly have nearly as many examples of passes that are very likely to fail as there are examples of passes that are very likely to succeed. This bias is reflected in our model performance metrics in tables 6.1, 6.5 and 6.6 as the precision of the negative class predictions is always significantly behind the precision of the positive class predictions. Clearly, there are no features that decisively identify very improbable passes.

There is one major shortcoming in our work on pass probabilities. By focusing on pass outcome we've now modelled the risk of passes,but if we only evaluate passes based on the associated risks involved the right answer is to always play backwards towards the own defenders and goalkeeper. Unfortunately, football is a game where the goal is to score goals in the opposing team's goal and for this risks have to be taken. Having a model for pass reward alongside the current pass successfulness model would let us accurately model the risk-reward tradeoff that is so important in an actual football match, significantly increasing the value of our work.

For future work, a basic approach for modelling pass reward would be taking our existing pass dataset, creating a subset of only the successful passes and adding a new label to these passes that describes whether a certain desirable event happened shortly after the ball arrived at the receiver, such as a shot on the goal. The authors of Ref. [41] used exactly this approach and we acknowledge that their work is further in this. The process of generating this second model and analyzing it, as well as combining the existing model and the new model for a proper risk-reward model, would generate enough new questions and avenues for analysis for it to merit its own research.

Chapter 8

Use cases

A lot of time has now been spent talking about the classification performance of the model, the importance of individual features, and the probabilistic output of the model, but relatively little time has been spent talking about the use cases for this model within the sports domain. In this chapter, several potential use cases for the probabilistic output of the model will be described alongside any avenues for future improvement.

8.1 Analyzing player decision making

Now that we've analyzed what kind of probabilities our model outputs for passes that are in our pass dataset, we will look at a use case for our model involving hypothetical passes that aren't in our dataset. So far we've looked at the probabilities of passes and how they compare to the actual outcome of said passes, but there is another dimension that can be touched upon by looking at *what-if* scenarios.

For every moment in a football match, the player that currently has ball possession has up to 10 teammates he or she can pass the ball to. We can use our feature engineering pipeline to calculate the mathematical features described so far for any of these moments by picking one of these 10 teammates as the intended receiver of a hypothetical pass. We can then use our logistic regression model to predict the probability that this hypothetical pass from the player in ball possession to the picked intended receiver would actually be successful. Repeating this process 10 times for every teammate on the field at that moment gives us the probabilities of all pass options at that moment and these probabilities can be used to compare the risk of different pass options.

A specific subset of moments for which such an analysis can be done is the moments at which an actual pass is about to start. At these moments, an intended receiver was chosen by the player in ball possession and an actual pass was about to be made. We can use the procedure just described to calculate probabilities for both the actual pass and the nine other alternative options that were available at the time and compare the player's decision to the options that were available. The result can then be used to evaluate a player's decision making in the field. An example of such an analysis is shown in figure 8.1. The actual intended receiver that was picked by the pass sender is highlighted and the probabilities for the pass towards him as well as any alternatives are shown. We've taken to calling this kind of visualization a *spider plot*, since the drawn (hypothetical) pass vectors remind us of a spider web with the pass sender at the center of the web.

As can be seen, the pass sender picked a relatively safe pass option back towards his

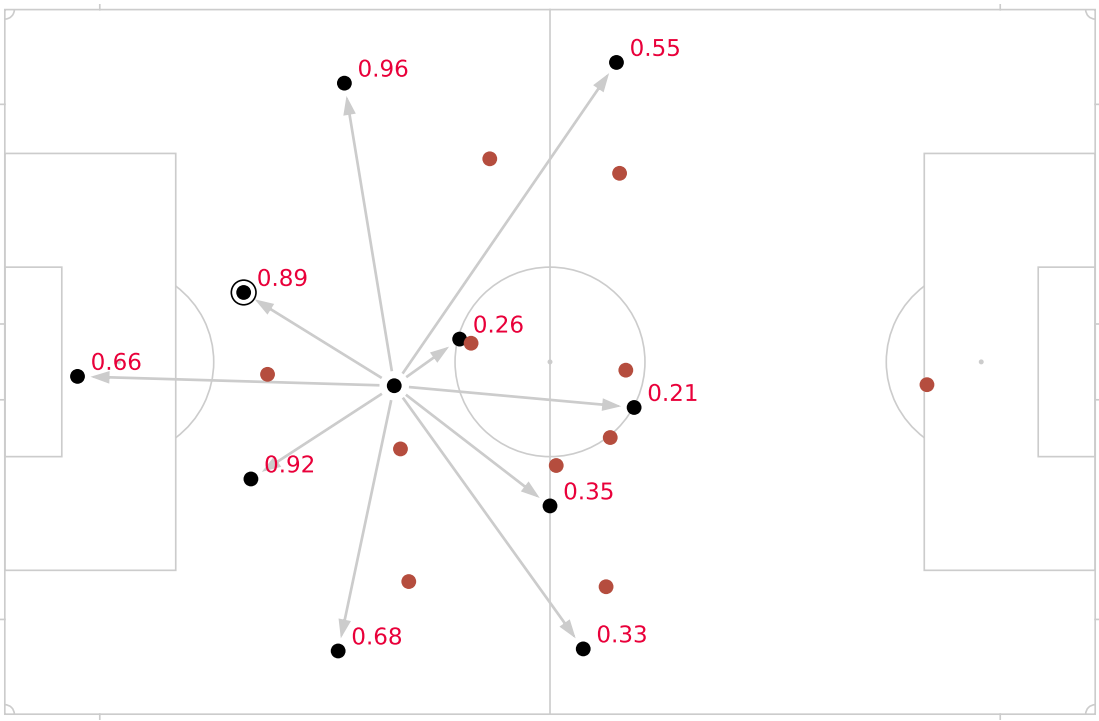


Figure 8.1: Predicted probability of success of a pass (actual receiver highlighted) and predicted probabilities of all nine alternative receivers at the moment the pass started

own goal line. A number of different pass options were available to the player, including a few the pass model deemed much riskier due to the proximity of opponents. It can also be seen that a few of the riskier alternatives available had an interesting tradeoff between risk and reward, such as the option of passing to the left forward in the top right of the plot. This pass still had a probability greater than 50% attached to it and such a pass, if successful, would've opened up the opportunity to launch an offensive towards the opposing goal line that could've resulted in a cross towards the center forward with the promise of a shot on the goal. The probabilities that are given by the model reflect what we would expect after looking at the probability distribution over the passes in our dataset. There are passes that are judged to be very unlikely to fail, such as the pass back to the back right defender, while there are no passes that the model judges to be impossible to perform successfully.

We believe that an analysis like this can tell players, coaches and sports analysts alike a lot about the decision making process of individual players and how well a situation was assessed by said players. Coaches can use it to reflect on the performance of players and give players feedback on the passing decisions they made in specific critical moments of a footbal match or training. Players can use it to better understand the decisions they made and the impact, or lack of impact, they had on the bigger picture. Finally, sport analysts can use the analysis to quantify the value of passes on a larger scale and aggregate the results for the team or particular players as well as highlighting outliers for deeper discussion. We came up with the idea of using these collections of probabilities to directly quantify how good the decision of a pass sender was. However, the lack of pass reward modelling is a severe limitation and an obvious improvement point in future work.

8.2 Labelling passes for player profiling

We've now analyzed how the probabilistic output from our model can be used to compare passes and their alternatives. In this subsection, we'll take a look at another use case which is directly using the probabilities to label passes for the purpose of analyzing the behavioral patterns of players.

The probabilistic output of our model can be thought of as being yet another label for the dataset, not unlike the actual pass successfulness label that we calculated in chapter 4 and have been using as the outcome variable throughout all the modelling done in this work. In doing so, we can now compare our passes along 2 dimensions: actual pass successfulness and the modelled probability of success. Naturally one would expect the 2 dimensions to be heavily correlated, but there may be outliers that we can detect in the process of doing this comparison.

In particular, we can label the passes in the dataset with probabilities and then subset the data per player to create a "risk profile" of a particular player. To illustrate, we will draw the pass risk profiles of a couple of players of the one team we have the most data of. Their profiles, based on 5-6 matches per player, are drawn in figure 8.2. The plots highlight the relation, for a particular player, between probability of pass success and the number of passes in the dataset as well as the proportion of pass successfulness.

We can see that every role seems to have its own distinct distribution. The goalkeeper in question has a distribution that is mostly dominated by very safe passes. The defender also emphasizes passes with a high probability of success, but has a much more balanced distribution containing many passes with a lower probability. Meanwhile, the midfielder has the most balanced distribution with even more low probability passes and the distribution of the forward is completely reversed, with an emphasis on more dangerous passes. All of the trends exhibited, match up with our expectations of what each role is supposed to be doing in a typical football match.

This type of plot can be used to assess how effectively a certain player handles very risky or very safe passes and whether such performance matches up with the expectations for the role that player is filling. For example, the defender and midfield shown above almost never fail at passes that fall within the 90-100% probability bracket. Meanwhile, the forward seems to have a slightly higher failure rate within that bracket. Players that exhibit unusual behaviour within their roles compared to what was shown here may have an edge over some of their fellow players in similar roles. Such plots could thus be used to assess how effective (and thus valuable) a player is in his or her particular role.

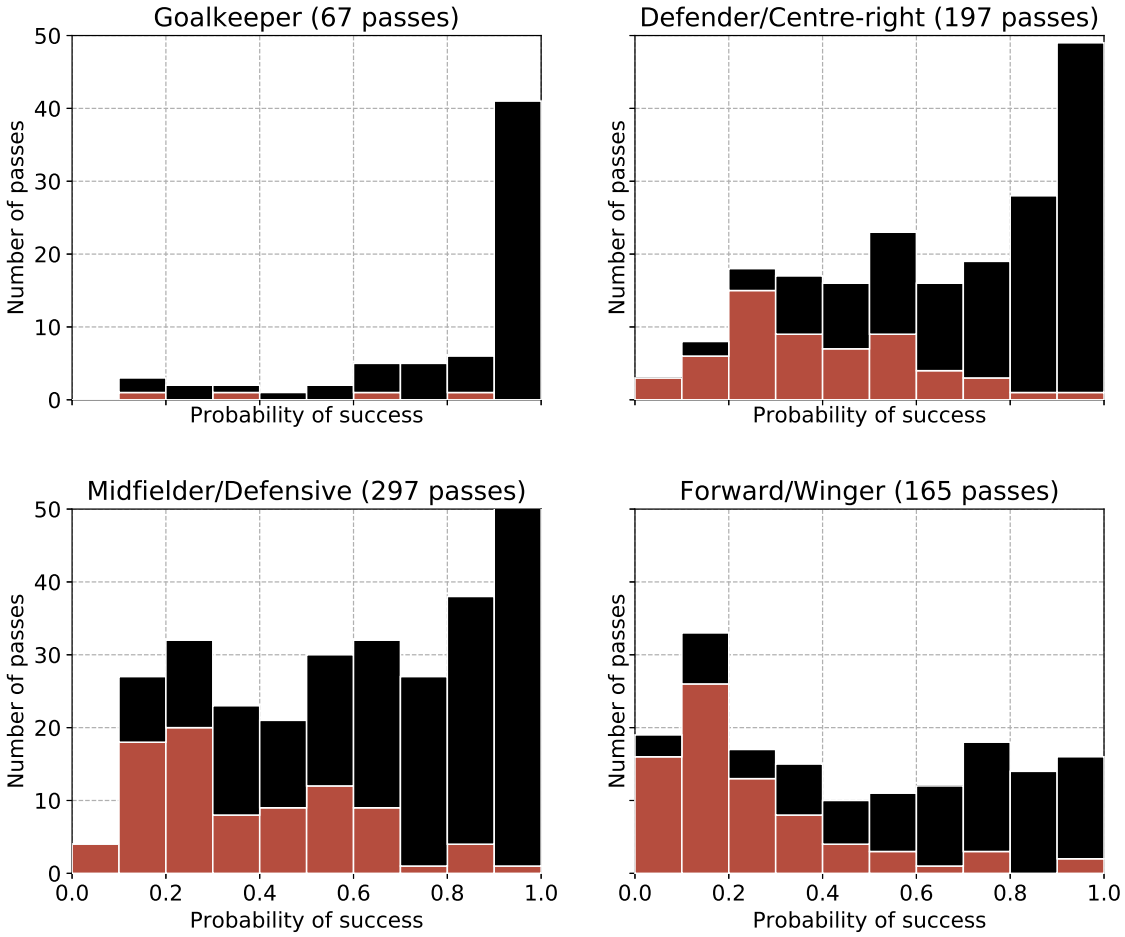


Figure 8.2: Player pass risk profiles of four different players in different roles. A player's pass risk profile is a histogram binning all the passes performed by said player by the probability of success our model assigns to each pass. Successful passes (black) and unsuccessful passes (red) are shown separately in a stacked fashion

Notice that the amount of data we currently have is relatively limited. With the size of our dataset, subsetting the data to only one particular player results in a very small subset of a few hundred passes at best. We believe that for best results, a bigger dataset would be needed that contains thousands of passes for a particular player. Due to the small sample size, the results shown here can exhibit high variance and we should not read too much into the highlighted trends. The results could be improved greatly with the availability of more data.

8.3 Labelling passes for outlier detection

Another interesting use case involving pass labelling is outlier detection. By relating the actual successfulness of passes to the probabilities our model assigns to said passes we can identify particularly significant outliers that tell us whether a player performed very well or very poorly in said situations. For example, focusing on the passes made by the forward in figure 8.2, we can look at all the passes that failed in the highest probability bracket as well as the passes that succeeded in the lowest probability bracket.

In figure 8.3, we've shown one of the 2 failed passes from the highest probability

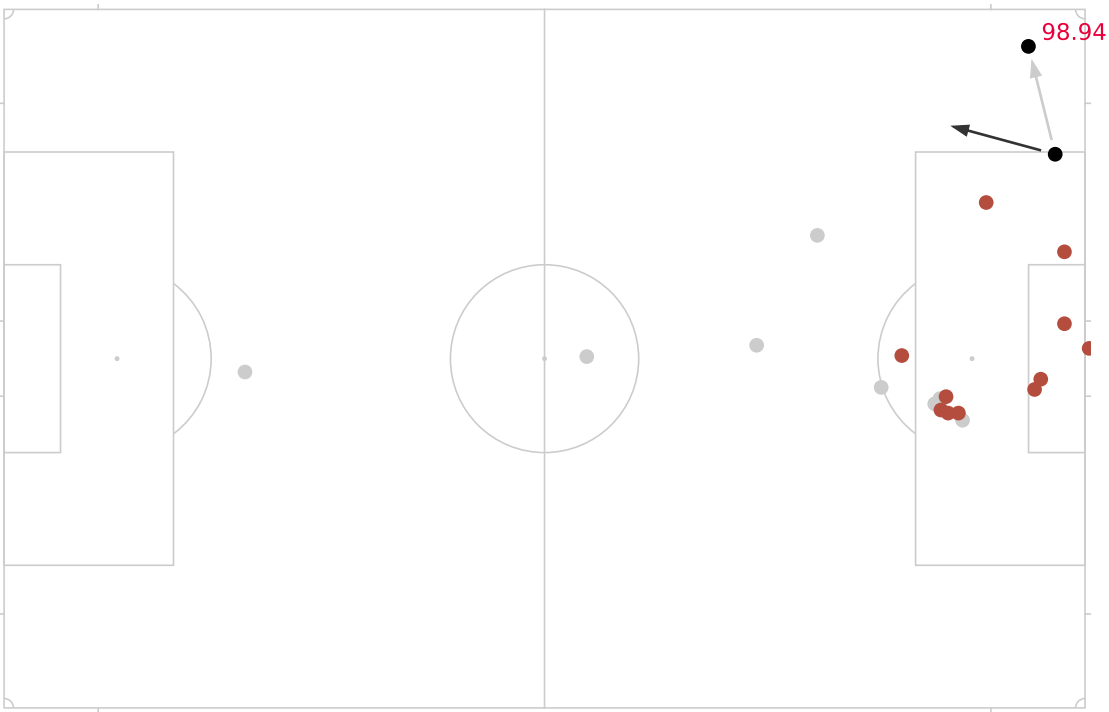


Figure 8.3: Pass that failed with high probability of success. The pass sender and intended receiver are shown (in black), as well as all the opponents on the field (in red). The vector between the sender and intended receiver is shown in light gray while the actual vector travelled by the ball is shown in black

bracket. The gray arrow marks the pass vector from the pass sender to intended receiver while the black arrow marks the actual path travelled by the ball within the next 1.5 seconds. We can see that while the pass vector was assigned a probability of almost 0.99, the pass sender seemed to have handled the pass poorly and actually sent the ball in the wrong direction, where an opponent was able to intercept it. This is a clear example of an anomaly caused by player error, and is something the model cannot account for given the information in the currently available position data.

As it turns out, passes that failed due to player error are often exactly the passes that appear to fail with high probabilities. As such, this method of analysis can be used to detect passes that were heavily affected by player error and these pass examples can then be used for direct post-match feedback to a player. Failed passes with a high probability of success can typically results in very negative outcomes for the team as a whole - typically they're passes close to the team's own goal and if an opponent gains possession of the ball at such a position, it leaves the team highly vulnerable to an opposing goal. Being able to quickly identify and label such passes when given the complete post-match dataset helps sports data analysts speed up their work and get insights related to these passes back to the coach and players much faster.

Similarly, figure 8.4 highlights one of the 2 passes that succeeded with a high probability of success. We can see that the pass had a mere $\sim$0.09 probability of success and is thus marked as a particularly risky pass, the pass actually succeeded and ended up putting a teammate in ball possession while very close to the opposing goal. This pass was actually a cross - a pass part of a typical attack pattern where a forward winger passes the ball from the side of the field directly to the center forward close to the goal

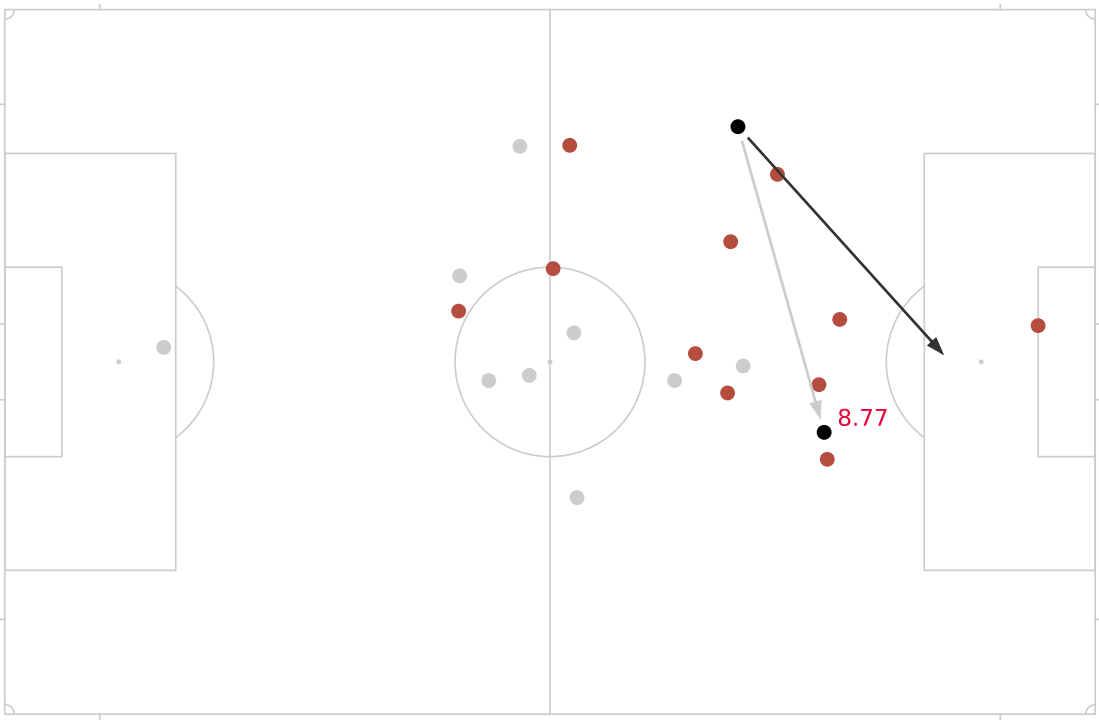


Figure 8.4: Pass that succeeded with low probability of success. The pass sender and intended receiver are shown (in black), as well as all the opponents on the field (in red). The vector between the sender and intended receiver is shown in light gray while the actual vector travelled by the ball is shown in black

in order to try and create goal scoring opportunities. Such crosses are often chipped (sent in an arc) to avoid opposing players being able to intercept in the middle of the pass vector.

We found that successful low probability passes like this one are typically the ones that generate actual goal scoring attempts for the team in question. Risks have to be taken in order to score goals, and said goals can typically be traced back to passes that had a significant chance to fail. As such, this analysis is very well suited to mining a post-match dataset for such decisive moments and it is relatively easy to link these decisive moments back to the initiating players.

The procedures for creating player profiles and finding positive and negative outliers discussed in the previous subsection and this subsection can be combined into an overall report for per-player performance, containing clear improvement points for the player, with examples of situations that should've been handled better, as well as positive feedback with examples of situations where a player performed very well.

Chapter 9

Conclusion

In this work, we offered a detailed description of a machine learning pipeline which is used to generate probabilistic models for the prediction of pass outcomes in a football match. This is a binary classification problem in which a pass is either successful, with the ball arriving at the intended receiver, or unsuccessful, with the ball not arriving at a team member or the match being interrupted before anyone receives it. One of the main benefits of such a probabilistic model is the availabilty of probabilistic output which can be interpreted as a confidence score for the prediction.

One of the questions we set out to answer at the start of this project was "What are the most important factors determining pass success in a football match?". In order to facilitate finding an answer to this question, we described the individual steps in our work from the dataset preparation all the way to the training and assessment of machine learning models. In particular, we gave a very detailed description of the feature engineering work done, describing every numerical feature calculated and the mathematical steps that were taken (based on available data) to generate them. In the analysis of our results, we then allocated significant time to describing the performance of every feature and how well they performed as part of the full model as well as in individual bivariate and trivariate models.

As it turns out, features defining a form of pressure which opponents are exerting on the general area around the pass vector - the path the ball traverses - perform very well on their own, with an accuracy that already reaches 0.724 compared to the baseline of 0.68. The complete model trained on all features went up to 0.744 accuracy, showing us the other features do not add nearly as much predictive power in terms of raw accuracy. Meanwhile, a feature describing how far away opponents are from the receiver helps the model identify passes that have a very high probability of success and thus greatly improve the precision of positive predictions, increasing it from a baseline precision of 0.68 to a precision of 0.864. In general, it seems that features describing the spatial interaction between the intended receiver of the pass and the players on the opposing team have very decisive predictive power. Similarly, features describing the spatial interaction between the pass vector and the positions of opposing players are also very strong features.

We also noticed a strong predictive trend in features describing the nature of the pass, such as its direction and distance while most of the other features describing the state of the players involved in the pass, such as the speed, direction and position of the players, seem to either correlate heavily with the aforementioned stronger features or seem to not have any predictive power. In hindsight, a lot of this makes sense and

matches our intuition - if the intended receiver and the pass vector are not covered by opponents, there is very little that can interfere with the successfulness of the pass besides player error. There are secondary features that synergize very well, such as the pass distance. Intuitively this makes sense, as opponents have more time to close the gap if it takes longer for the ball to reach the intended receiver.

The second question we wanted to answer was "How does a predictive model built on these factors add value for the stakeholders involved?". Our model's predictive potential for the binary classification of pass successfulness is significantly better than a naive baseline model and can be used to assess whether a pass is likely to succeed or fail. We consider the precision of predictions of pass success to be the most important metric for a model like this, as the cost of losing the ball when this is not expected can have dire consequences in a football match. We are happy to report that our model achieves a precision of 0.88 on the positive class, which is much better than a naive baseline model does.

Classification performance is only one aspect of the model we described in our work. Due to our choice of logistic regression as machine learning algorithm, we have created a naturally probabilistic model which bases its classification results on probabilistic output. Such probabilities can be used as confidence scores for the aforementioned predictions and opens up exciting new use cases as well. Some of the use cases that we mentioned in our results analysis include the following:

- Analyzing the probabilities of a pass and the available alternatives at that moment, in order to quantify player decision making skills
- Labelling a dataset of passes with the probabilities assigned by the model, enabling sports analysts to compare probabilities to actual outcomes and easily find outliers

While there are several shortcomings that still need to be addressed in future work, our model enables sports data analysts to mine their datasets for pass-related insights much faster than before while also letting them provide the players and coach with a completely new perspective on how to value the quality of passes and player decisions in a football match.

Bibliography

[1] Richard Akenhead and George Nassis. Training load and player monitoring in high-level football: Current practice and perceptions. *International Journal of Sports Physiology and Performance*, 10 2015.

[2] Gennady Andrienko, Natalia Andrienko, Guido Budziak, Tatiana Landesberger, and Hendrik Weber. Exploring pressure in football. pages 1–3, 05 2018.

[3] Abdallah Bashir. Comparative study on classification performance between support vector machine and logistic regression. *International Journal of Machine Learning and Cybernetics*, 4, 02 2012.

[4] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, February 2012.

[6] Viv Bewick, Liz Cheek, and Jonathan Ball. Statistics review 7: Correlation and regression. *Critical care (London, England)*, 7:451–9, 01 2004.

[7] Viv Bewick, Liz Cheek, and Jonathan Ball. Statistics review 13: Receiver operating characteristic curves. *Critical care (London, England)*, 8:508–12, 01 2005.

[8] Viv Bewick, Liz Cheek, and Jonathan Ball. Statistics review 14: Logistic regression. *Critical care (London, England)*, 9:112–8, 03 2005.

[9] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. Large-scale analysis of soccer matches using spatiotemporal tracking data. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2015:725–730, 01 2015.

[10] Joel Brooks, Matthew Kerr, and John Guttag. Developing a data-driven player ranking in soccer using predictive model weights. pages 49–55, 08 2016.

[11] Gavin Cawley and Nicola Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 07 2010.

[12] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 01 2002.

[13] Sanjay Chawla, Joël Estephan, Joachim Gudmundsson, and Michael Horton. Classification of passes in football matches using spatiotemporal data. *ACM Trans. Spatial Algorithms Syst.*, 3(2):6:1–6:30, August 2017.

[14] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. 02 2015.

[15] Christian Collet. The possession game? a comparative analysis of ball retention and team success in european and international football, 20072010. *Journal of sports sciences*, 31, 10 2012.

[16] The Business Research Company. Sports global market opportunities and strategies to 2022. `https://www.researchandmarkets.com/reports/4770417/sports-global-market-opportunities-and-strategies`, May 2019. Accessed: 2020-01-25.

[17] Raphael Couronn, Philipp Probst, and Anne-Laure Boulesteix. Random forest versus logistic regression: A large-scale benchmark experiment. *BMC Bioinformatics*, 19, 12 2018.

[18] J.S. Cramer. The origins of logistic regression. *Tinbergen Institute, Tinbergen Institute Discussion Papers*, 01 2002.

[19] Adele Cutler, David Cutler, and John Stevens. *Random Forests*, volume 45, pages 157–176. 01 2011.

[20] Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 01 2001.

[21] Wouter Frencken and Koen A.P.M. Lemmink. *Team kinematics of small-sided soccer games: A systematic approach*, pages 161–166. 01 2009.

[22] Leilani Gilpin, David Bau, Ben Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. pages 80–89, 10 2018.

[23] Floris Goes, Matthias Kempe, Rens Meerhoff, and Koen A.P.M. Lemmink. Not every pass can be an assist: A data-driven model to measure pass effectiveness in professional soccer matches. *Big Data*, 7, 09 2018.

[24] Pablo M Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi. Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems*, 83(2):83–90, 2006.

[25] Deloitte Sports Business Group. European football market worth €28.4 billion (£25.1bn) as premier league clubs lead the way to record revenues. `https://www2.deloitte.com/uk/en/pages/press-releases/articles/european-football-market-worth-28-billion-euros-as-premier-league-clubs-lead-the-way-to-record-revenues.html`, May 2019. Accessed: 2020-01-25.

[26] Laszlo Gyarmati and Xavier Anguera. Automatic extraction of the passing strategies of soccer teams. 08 2015.

[27] Trevor Hastie, Rob Tibshirani, and Jerome Friedman. *Model Assessment and Selection*, pages 219–259. 08 2009.

[28] Sreenivasa Jammalamadaka and Ashis Sengupta. *Topics in circular statistics*, volume 5. 01 2001.

[29] Daniel Link, Steffen Lang, and Philipp Seidenschwarz. Real time quantification of dangerousity in football using spatiotemporal tracking data. In *PloS one*, 2016.

[30] Patrick Lucey, Alina Bialkowski, Peter Carr, Eric Foote, and Iain Matthews. Characterizing multi-agent team behavior from partial team tracings : Evidence from the english premier league. volume 2, 01 2012.

[31] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, and Iain Matthews. quality vs quantity: Improved shot prediction in soccer using strategic features from spatiotemporal data. In *Proceedings of the 8th Annual MIT Sloan Sports Analytics Conference*, pages 1–9, 2015.

[32] Patrick Lucey, Dean Oliver, Peter Carr, Joe Roth, and Iain Matthews. Assessing team strategy using spatiotemporal data. pages 1366–1374, 08 2013.

[33] MarketsandMarkets. Sports technology market by technology, sports, and geography - global forecast to 2024. `https://www.marketsandmarkets.com/Market-Reports/sports-technology-market-104958738.html`, April 2019. Accessed: 2020-01-25.

[34] Felipe Moura, Luiz Eduardo Barreto Martins, and Sergio Cunha. Analysis of football game-related statistics using multivariate techniques. *Journal of sports sciences*, 32:1–7, 04 2014.

[35] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.

[36] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. pages 625–632, 01 2005.

[37] Joel Oberstone. Differentiating the top english premier league football clubs from the rest of the pack: Identifying the keys to success. *Journal of Quantitative Analysis in Sports*, 5:10–10, 01 2009.

[38] Hyeoun-Ae Park. An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing*, 43:154–164, 04 2013.

[39] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[40] Kedar Potdar, Taher Pardawala, and Chinmay Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, 175:7–9, 10 2017.

[41] Paul Power, Hector Ruiz, Xinyu Wei, and Patrick Lucey. Not all passes are created equal: Objectively measuring the risk and reward of passes in soccer from tracking data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1605–1613, New York, NY, USA, 2017. ACM.

[42] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of Database Systems*, 532538:532–538, 01 2009.

[43] Zuzana Reitermanová. Data splitting. 2010.

[44] Alice Richardson. Nonparametric statistics: A step-by-step approach. *International Statistical Review*, 83, 04 2015.

[45] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. In *PloS one*, 2015.

[46] Roque J. Saltarén, Garcia Cena, and Gutierrez de Abascal. On the velocity and acceleration estimation from discrete time-position sensors. 2015.

[47] Claude Sammut and Geoffrey I. Webb. *Encyclopedia of Machine Learning*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[48] F Sharifi, Vincent Hayward, and C.-S.J. Chen. Discrete-time adaptive windowing for velocity estimation. *Control Systems Technology, IEEE Transactions on*, 8:1003 – 1009, 12 2000.

[49] ukasz Szczepaski and I.G. Mchale. Beyond completion rate: Evaluating the passing ability of footballers. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 179, 03 2015.

[50] J.M. Vergara, Luis Esteban, Gerardo Sanz, ngel Fernando, and Javier Lopez. Logistic regression versus neural networks for medical data. *Ninth international conference Zaragoza-Pau on applied mathematics and statistics : Jaca (Spain), September 19-21, 2005, 2006-01-01, ISBN 84-7733-871-X, pags. 245-252*, 01 2006.

[51] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13:55–75, 08 2018.

[52] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–21, 01 2019.

Appendices

Appendix A

Amisco data description

This appendix contains a tabular overview of each of the 3 Amisco system datasets that we based our research on, which can be used to get a sense of the information contained within the data as well as its' quality. In addition to that, an overview is given of the types of events present in the event data.

A.1 Dataset columns

Column	Type	Description
PeriodId	Constant integer	The period of the match. Indicates the match half or overtime period
PlayerId	Integer	The unique identifier of a particular participant (player, referee or ball)
Time	Double	The number of seconds that has passed since the period start
X	Double	The X coordinate of the participant's position
Y	Double	The Y coordinate of the participant's position

Table A.1: Features in the Amisco position data

Column	Type	Description
BallEventCode	Categorical	Flag indicating what kind of ball event happened. Mutually exclusive with MatchEventCode
GoalZone	-	*Unused*
How	Categorical	Indicates whether the ball event was performed while running, jumping or sliding
Id	Integer	Unique identifier for the event
PeriodId	Constant integer	The period of the match
MatchEventCode	Categorical	Flag indicating what kind of match event happened. Mutually exclusive with BallEventCode
NbEvents	Constant integer	The total number of events in the period
PlayerIdJ1	Integer	Unique identifier of the primary player involved in the event
PlayerIdJ2	Integer	Unique identifier of the secondary player involved in the event
X	Double	The X coordinate at which the event took place
Y	Double	The Y coordinate at which the event took place
Result	Categorical	The outcome of the event, if any
Time	Double	The number of seconds that has passed since the period start
WithWhat	Categorical	Flag indicating which body part was used for a ball event

Table A.2: Features in the Amisco event data

Column	Type	Description
TeamCode	Categorical	Shorthand code for the team name of the participant
Color1	Integer	Primary decimal color code associated with this participant
Color2	Integer	Secondary decimal color code associated with this participant.
FirstName	String	First name of the participant
LastName	String	Last name of the participant
TeamId	Integer	Unique identifier of the participant's team. 0 for referees
TeamName	String	Full name of the participant's team. Unset for referees
PlayerId	Integer	Unique player identifier associated with this participant. Unset for coaches
Poste	Integer	Role identifier that describes the player's role within his/her team. Unset for coaches
RuleId	Integer	More high level identifier that describes the player's role within the match. 0 for players, 1 for coaches, 2 for referees
ShirtNumber	Integer	The number on the participant's shirt. Unset for coaches
ShoeColor	Constant integer	Decimal color code for the participant's shoes. Set to *black* for everyone
ShortColor	Constant integer	Decimal color code for the participant's shorts. Set to *white* for everyone
SocketColor	Constant integer	Another unused decimal color code that simply gets set to *white* for everyone

Table A.3: Features in the Amisco player metadata

A.2 Event types

Ball event code	Description
Empty	*Not a ball event*
Pass	An attempted pass from one player to a team member
Deep forward pass	A specific kind of pass where the ball is passed a lot more aggressively
Cross	A specific kind of pass where the ball is passed from the side of the field to the middle of the field, close to the opponent's goal
Reception	The ball is received by a player
Clearance	The ball is kicked away with no particular target
Neutral clearance	Same as the previous ball event, but unintentionally so
Shot on target	Shot on the goal
Shot not on target	Shot that was intended for the goal, but missed
Neutral contact	A player unintentionally influenced the direction of the ball
Running with ball	A player is moving around with the ball
Low catch drop gk	A goalkeeper dives and catches a ball travelling at low altitude
Low catch gk	A goalkeeper catches a ball travelling at low altitude
High catch drop gk	A goalkeeper dives and catches a ball travelling at higher altitude
High catch gk	A goalkeeper catches a ball travelling at higher altitude
Low deflection gk	A goalkeeper deflects a ball travelling at low altitude
High deflection gk	A goalkeeper deflects a ball travelling at higher altitude
Foot clearance gk	A goalkeeper kicks the ball away from the goal
Hold of ball gk	A goalkeeper grabs the ball

Table A.4: Possible BallEventCode values

Match event code	Description
Empty	*Not a match event*
Out for throw-in	The ball passes the side line
Out for corner	The ball passes the goal line after that side's team last touched it
Out for goal kick	The ball passes the goal line after the other side's team last touched it
Goal	A goal is scored
Own goal	A player directed the ball into their own team's goal
Yellow card	A player receives a yellow card
Foul - penalty	A player committed a foul and the other team is awarded a penalty as a result
Foul - dir. free-kick	A player committed a foul and the other team is awarded a free kick as a result
Interruption game	The match is interrupted for an unspecified reason
Player out	A player is moved out of the field by his coach
Player in	A player is placed on the field by his coach
Offside	The ball is passed to a team member who is standing behind the last defender of the other team
Goal post	The ball hits the goal post

Table A.5: Possible MatchEventCode values

Appendix B

Feature engineering overview

In the feature engineering chapter, all the mathematical features calculated for each pass are described. In this appendix, a tabular overview is given of all these features, including their names, descriptions and format.

A table has been added for each of the 4 main feature categories - *pass participants*, *pass characteristics*, *opponents' influence* and *contextual information*.

Name	Description	Units	Range	Identifier
Sender's speed	The absolute speed of the pass sender at the time of the pass	m/s	$[0, \infty >$	*speed_s*
Sender's direction	The direction of the pass sender at the time of the pass, relative to his team's direction of play	deg °	$[-90, 90]$	*direction_s*
Receiver's speed	The absolute speed of the *intended* pass receiver at the time of the pass	m/s	$[0, \infty >$	*speed_r*
Receiver's direction	The direction of the *intended* pass receiver at the time of the pass, relative to his team's direction of play	deg °	$[-90, 90]$	*direction_r*
Sender's relative X coordinate	The depth of the pass sender at the moment the pass starts	m	$< -\infty, \infty >$	*x_s*
Sender's absolute relative Y coordinate	The pass sender's distance from the middle line at the moment the pass starts	m	$[0, \infty >$	*y_s*
Receiver's relative X coordinate	The depth of the intended receiver at the moment the pass starts	m	$< -\infty, \infty >$	*x_r*
Receiver's absolute relative Y coordinate	The intended receiver's distance from the middle line at the moment the pass starts	m	$[0, \infty >$	*y_r*

Table B.1: Pass participant features

Name	Description	Units	Range	Identifier
Pass distance	The distance between the intended receiver and sender which has to be covered by the ball	m	$[0, \infty >$	*pass_distance*
Pass direction	The relative direction the ball has to be passed to, relative to the team's direction of play	deg °	$[-90, 90]$	*pass_direction*
Sender's pass angle	Angle between the direction the pass sender is moving and the direction the ball is being passed in	s	$[0, 180]$	*pass_angle_s*
Receiver's pass angle	Angle between the direction the intended receiver is moving and the direction the ball is being passed from	s	$[0, 180]$	*pass_angle_r*

Table B.2: Pass trajectory features

Name	Description	Units	Range	Identifier
Nearest opponent to sender	The distance between the pass sender and his/her nearest opponent	m	$[\,0, \infty >$	*opp_distance_s*
Velocity of nearest opponent towards sender	The nearest opponent's speed component projected directly towards the pass sender	m/s	$[\,0, \infty >$	*opp_velocity_s*
Nearest opponent to receiver	The distance between the pass receiver and his/her nearest opponent	m	$[\,0, \infty >$	*opp_distance_r*
Velocity of nearest opponent towards receiver	The nearest opponent's speed component projected directly towards the pass receiver	m/s	$[\,0, \infty >$	*opp_velocity_r*
Pass vector's free area	Minimum distance from any point on the pass vector to an opponent	m	$[\,0, \infty >$	*opp_distance_pass*
Pass vector's free angle	Minimum angle from the pass vector to any opponent vector (defined as the vector from the pass sender to the opponent)	deg °	$[\,0, 180\,]$	*opp_angle_pass*
Highest pressure on sender	Most pressuring opponent's pressure exerted on the pass sender	%	$[\,0, 100\,]$	*opp_pressure_s*
Total pressure on sender	Total pressure exerted on the pass sender by the entire opposing team	%	$[\,0, 100\,]$	*total_pressure_s*
Highest pressure on receiver	Most pressuring opponent's pressure exerted on the pass receiver	%	$[\,0, 1200\,]$	*opp_pressure_r*
Total pressure on receiver	Total pressure exerted on the pass receiver by the entire opposing team	%	$[\,0, 1200\,]$	*total_pressure_r*
Pressure on pass vector	Total amount of pressure exerted on the pass vector by the entire opposing team	%	$[\,0, 1100\,]$	*pass_pressure*

Table B.3: Opponents' influence features

Name	**Description**	**Units**	**Range**	**Identifier**
Player ball possession time	How long the pass sender has had control of the ball without interruption	s	$[0, \infty >$	*player_possession_t*
Team ball possession time	How long the pass sender's team has had control of the ball without interruption	s	$[0, \infty >$	*team_possession_t*
Pass technique	The part of the body used to pass the ball	Categorical	*chest* *header* *left_foot* *right_foot* *two_hands*	*pass_technique*
Passing team	The identity of the team passing the ball	Categorical	*team1* *team2* *team3* *team4* *team5* *team6*	*passing_team*
Sender role	The role associated with the pass sender	Categorical	*goalkeeper* *defender* *midfield* *forward*	*sender_role*
Receiver role	The role associated with the intended receiver	Categorical	*goalkeeper* *defender* *midfield* *forward*	*receiver_role*

Table B.4: Contextual information features

Appendix C

Model assessment/fitting reports

In this appendix the raw terminal-based output from the predictive modelling component of our pipeline can be found. A section is present for every model that has been trained, with its' relevant pipeline output being included there.

The listed output details information about the nested cross validation procedure used for model performance assessment, such as the optimal hyperparameters and test accuracy for the train-test split of each iteration of the outer cross-validation loop. The performance across all test folds is aggregated into a report that lists the average test accuracy, a confusion matrix and a classification report listing many useful metrics such as precision, recall and f1-scores. These metrics can be used to directly compare the merits of a model.

At the end of each report a small description of the final model fitting procedure is given as well, detailing the optimal hyperparameters found using cross validation over the entire dataset. No test scores are given here as the final model isn't tested on an independent dataset at all. The performance assessment of nested cross-validation is instead used as an estimation of the generalization error.

```
------------------------------------------------------------
Generalization error estimation using 10-fold nested Cross-Validation
Classifier: LogisticRegression
Dataset size: 4107
Hyperparameters to optimize: ['C']
------------------------------------------------------------
Fold 1 test error: 0.7640 ( C=497.7023564332114 )
Fold 2 test error: 0.7713 ( C=869.7490026177834 )
Fold 3 test error: 0.7470 ( C=0.07564633275546291 )
Fold 4 test error: 0.7445 ( C=20.09233002565046 )
Fold 5 test error: 0.7543 ( C=61.35907273413176 )
Fold 6 test error: 0.7153 ( C=0.1747528400007685 )
Fold 7 test error: 0.7299 ( C=0.23101297000831603 )
Fold 8 test error: 0.7341 ( C=0.1747528400007685 )
Fold 9 test error: 0.7512 ( C=6.5793322465756825 )
Fold 10 test error: 0.7268 ( C=0.049770235643321115 )
------------------------------------------------------------
Nested Cross-Validation finished in 818.0 seconds. Calculating metrics..
------------------------------------------------------------
Average test error: 0.7438

Confusion matrix
Prediction         1         0     Total

  Class: 1      2029       763      2792
  Class: 0       289      1026      1315

     Total      2318      1789      4107

Classification report
              precision    recall  f1-score   support

           1       0.88      0.73      0.79      2792
           0       0.57      0.78      0.66      1315

    accuracy                           0.74      4107

------------------------------------------------------------
Model training using 10-fold Cross-Validation
Classifier: LogisticRegression
Dataset size: 4107
Hyperparameters to optimize: ['C']
------------------------------------------------------------
Hyperparameters found: C=4.9770235643321135
------------------------------------------------------------
Hyperparameter optimization and model fitting finished in 96.3 seconds
```

```
------------------------------------------------------------
Generalization error estimation using 10-fold nested Cross-Validation
Classifier: AdaBoostClassifier
Dataset size: 4107
Hyperparameters to optimize: ['n_estimators', 'learning_rate']
------------------------------------------------------------
Fold 1 test error: 0.7324 ( n_estimators=70 learning_rate=0.6300000000000001 )
Fold 2 test error: 0.7664 ( n_estimators=70 learning_rate=0.5900000000000001 )
Fold 3 test error: 0.7640 ( n_estimators=70 learning_rate=1.0300000000000005 )
Fold 4 test error: 0.7543 ( n_estimators=70 learning_rate=0.5700000000000001 )
Fold 5 test error: 0.7324 ( n_estimators=70 learning_rate=0.9000000000000004 )
Fold 6 test error: 0.7178 ( n_estimators=70 learning_rate=0.56 )
Fold 7 test error: 0.7324 ( n_estimators=70 learning_rate=1.4200000000000008 )
Fold 8 test error: 0.7122 ( n_estimators=70 learning_rate=0.8900000000000003 )
Fold 9 test error: 0.7512 ( n_estimators=70 learning_rate=1.3000000000000007 )
Fold 10 test error: 0.7488 ( n_estimators=70 learning_rate=1.0000000000000004 )
------------------------------------------------------------
Nested Cross-Validation finished in 1755.8 seconds. Calculating metrics..
------------------------------------------------------------
Average test error: 0.7412

Confusion matrix
Prediction         1         0     Total

  Class: 1      2066       726      2792
  Class: 0       337       978      1315

     Total      2403      1704      4107

Classification report
              precision    recall  f1-score   support

           1       0.86      0.74      0.80      2792
           0       0.57      0.74      0.65      1315

    accuracy                           0.74      4107
   macro avg       0.72      0.74      0.72      4107
weighted avg       0.77      0.74      0.75      4107

------------------------------------------------------------
Model training using 10-fold Cross-Validation
Classifier: AdaBoostClassifier
Dataset size: 4107
Hyperparameters to optimize: ['n_estimators', 'learning_rate']
------------------------------------------------------------
Hyperparameters found: n_estimators=70 learning_rate=0.9800000000000004
------------------------------------------------------------
Hyperparameter optimization and model fitting finished in 72.6 seconds
```

```
------------------------------------------------------------
Generalization error estimation using 10-fold nested Cross-Validation
Classifier: RandomForestClassifier
Dataset size: 4107
Hyperparameters to optimize: ['n_estimators', 'max_depth']
------------------------------------------------------------
Fold 1 test error: 0.7689 ( n_estimators=150 max_depth=19 )
Fold 2 test error: 0.7567 ( n_estimators=150 max_depth=25 )
Fold 3 test error: 0.7470 ( n_estimators=150 max_depth=22 )
Fold 4 test error: 0.7226 ( n_estimators=150 max_depth=20 )
Fold 5 test error: 0.7981 ( n_estimators=150 max_depth=None )
Fold 6 test error: 0.7324 ( n_estimators=150 max_depth=23 )
Fold 7 test error: 0.7616 ( n_estimators=150 max_depth=20 )
Fold 8 test error: 0.7537 ( n_estimators=150 max_depth=22 )
Fold 9 test error: 0.7732 ( n_estimators=150 max_depth=21 )
Fold 10 test error: 0.7537 ( n_estimators=150 max_depth=19 )
------------------------------------------------------------
Nested Cross-Validation finished in 2541.8 seconds. Calculating metrics..
------------------------------------------------------------
Average test error: 0.7568

Confusion matrix
Prediction         1         0     Total

  Class: 1      2293       499      2792
  Class: 0       500       815      1315

     Total      2793      1314      4107

Classification report
              precision    recall  f1-score   support

           1       0.82      0.82      0.82      2792
           0       0.62      0.62      0.62      1315

    accuracy                           0.76      4107
   macro avg       0.72      0.72      0.72      4107
weighted avg       0.76      0.76      0.76      4107

------------------------------------------------------------
Model training using 10-fold Cross-Validation
Classifier: RandomForestClassifier
Dataset size: 4107
Hyperparameters to optimize: ['n_estimators', 'max_depth']
------------------------------------------------------------
Hyperparameters found: n_estimators=150 max_depth=22
------------------------------------------------------------
Hyperparameter optimization and model fitting finished in 291.6 seconds
```

```
------------------------------------------------------------
Generalization error estimation using 10-fold nested Cross-Validation
Classifier: SVC
Dataset size: 4107
Hyperparameters to optimize: ['C', 'kernel']
------------------------------------------------------------
Fold 1 test error: 0.7664 ( C=1.0 kernel=rbf )
Fold 2 test error: 0.7202 ( C=1.0 kernel=poly )
Fold 3 test error: 0.7372 ( C=0.5 kernel=poly )
Fold 4 test error: 0.7762 ( C=0.5 kernel=poly )
Fold 5 test error: 0.7543 ( C=0.5 kernel=rbf )
Fold 6 test error: 0.7105 ( C=1.0 kernel=linear )
Fold 7 test error: 0.7348 ( C=0.5 kernel=rbf )
Fold 8 test error: 0.7244 ( C=0.5 kernel=poly )
Fold 9 test error: 0.7390 ( C=0.5 kernel=poly )
Fold 10 test error: 0.7439 ( C=0.5 kernel=poly )
------------------------------------------------------------
Nested Cross-Validation finished in 1323.0 seconds. Calculating metrics..
------------------------------------------------------------
Average test error: 0.7407

Confusion matrix
Prediction         1         0     Total

  Class: 1      2054       738      2792
  Class: 0       327       988      1315

     Total      2381      1726      4107

Classification report
              precision    recall  f1-score   support

           1       0.86      0.74      0.79      2792
           0       0.57      0.75      0.65      1315

    accuracy                           0.74      4107
   macro avg       0.72      0.74      0.72      4107
weighted avg       0.77      0.74      0.75      4107

------------------------------------------------------------
Model training using 10-fold Cross-Validation
Classifier: SVC
Dataset size: 4107
Hyperparameters to optimize: ['C', 'kernel']
------------------------------------------------------------
Hyperparameters found: C=0.5 kernel=poly
------------------------------------------------------------
Hyperparameter optimization and model fitting finished in 164.1 seconds
```

```
------------------------------------------------------------
Generalization error estimation using 10-fold nested Cross-Validation
Classifier: KNeighborsClassifier
Dataset size: 4107
Hyperparameters to optimize: ['n_neighbors']
------------------------------------------------------------
Fold 1 test error: 0.7251 ( n_neighbors=9 )
Fold 2 test error: 0.6959 ( n_neighbors=11 )
Fold 3 test error: 0.7007 ( n_neighbors=13 )
Fold 4 test error: 0.7372 ( n_neighbors=13 )
Fold 5 test error: 0.6618 ( n_neighbors=13 )
Fold 6 test error: 0.7324 ( n_neighbors=13 )
Fold 7 test error: 0.7299 ( n_neighbors=13 )
Fold 8 test error: 0.6951 ( n_neighbors=13 )
Fold 9 test error: 0.6976 ( n_neighbors=13 )
Fold 10 test error: 0.7390 ( n_neighbors=13 )
------------------------------------------------------------
Nested Cross-Validation finished in 203.3 seconds. Calculating metrics..
------------------------------------------------------------
Average test error: 0.7115

Confusion matrix
Prediction         1         0     Total

  Class: 1      1957       835      2792
  Class: 0       350       965      1315

     Total      2307      1800      4107

Classification report
              precision    recall  f1-score   support

           1       0.85      0.70      0.77      2792
           0       0.54      0.73      0.62      1315

    accuracy                           0.71      4107
   macro avg       0.69      0.72      0.69      4107
weighted avg       0.75      0.71      0.72      4107

------------------------------------------------------------
Model training using 10-fold Cross-Validation
Classifier: KNeighborsClassifier
Dataset size: 4107
Hyperparameters to optimize: ['n_neighbors']
------------------------------------------------------------
Hyperparameters found: n_neighbors=13
------------------------------------------------------------
Hyperparameter optimization and model fitting finished in 25.1 seconds
```