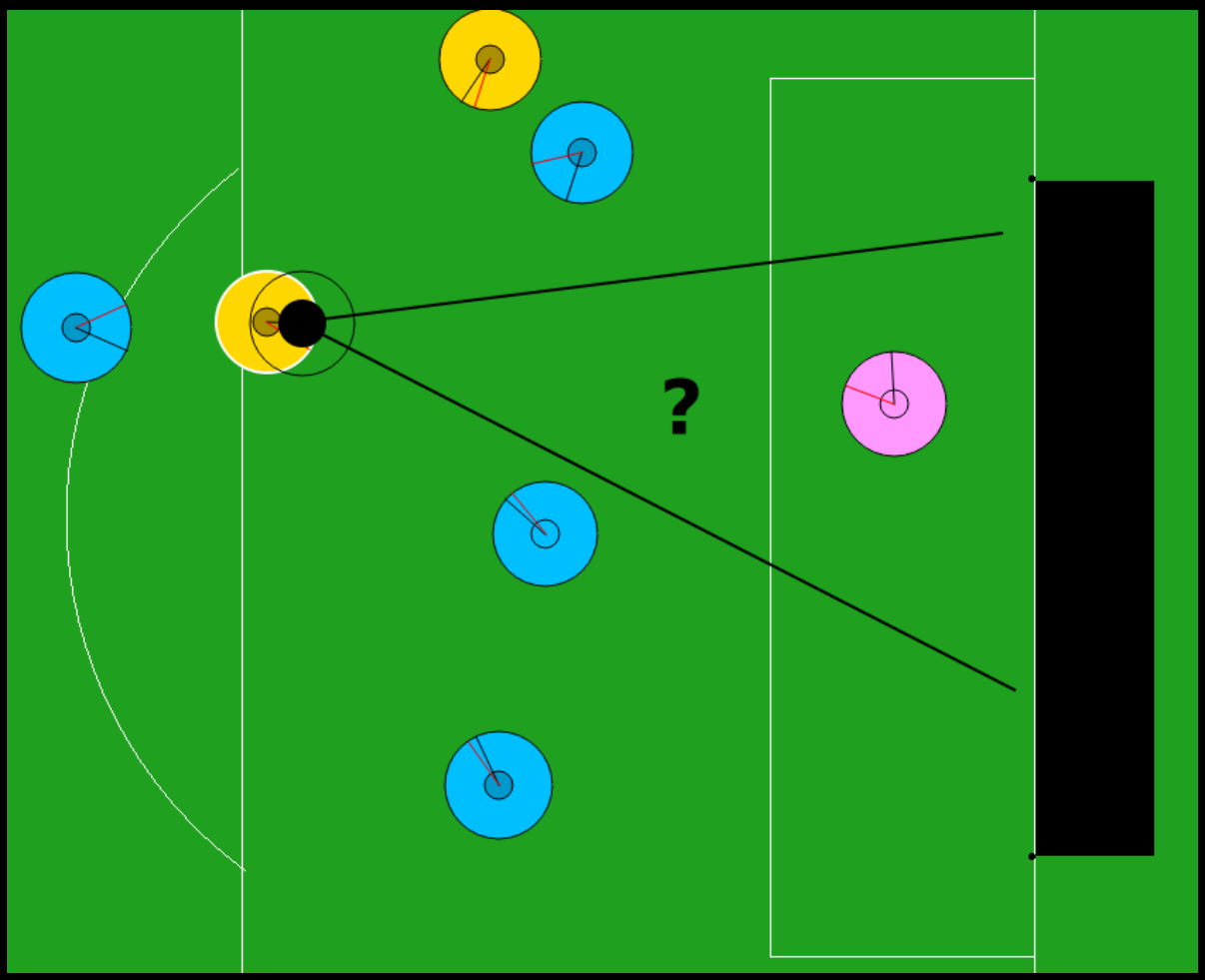
Soccer behaviour prediction with deep neural networks in Soccer Simulation 2D



Daniël van Dijk

Layout: typeset by the author using LaTeX.
Cover illustration: own illustration, made with the SS2D soccer monitor.

Soccer behaviour prediction with deep neural networks in Soccer Simulation 2D

Daniël van Dijk
12697834

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Semester 2, 2022

Abstract

Predicting the offensive soccer behaviours (passing, shooting and dribbling) is an active research area in the field of robotic soccer. Zare et al. modelled pass behaviour by predicting the intended receiver of a pass with deep neural networks in the Soccer Simulation 2D environment. The intended target player was predicted with 84% accuracy, which validated the effectiveness of their taken approach for pass behaviour classification. However, this approach for behaviour modelling can also be validated with regression models. Therefore, this thesis investigates how accurate deep neural networks can predict agents' passing, shooting and dribbling behaviour in a Soccer Simulation 2D match when shifting from classification models to regression models. Furthermore, the importance of features for the models' predictions of shooting and dribbling behaviour is analysed.

The shooting and dribbling behaviours of soccer agents can be predicted accurately when the training data is generated without noise in the observations and the same team is used as opponent. An adjusted R^2-score of 93% is reached with the best shot prediction model, and an average adjusted score of 94% is reached with the best dribble prediction model. These R^2-scores improved the baseline performance with 7% and 5%, respectively. The pass regression model reached an average adjusted R^2-score of 88% but did not improve the baseline.

Acknowledgements

I would like to thank my supervisor, Dr. Arnoud Visser, for his guidance during this project. In addition, I would like to thank the CYRUS team for developing the Data Extractor module and making it publicly available. Finally, I would like to thank Nader Zare, member of the CYRUS team, for answering my questions regarding the software.

Contents

Chapter 1

Introduction

1.1 Background

"By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup" (Kitano & Asada 1998). This highly ambitious goal of the RoboCup research initiative, which uses soccer to advance the development of AI and robotics, was initially inspired by the Apollo space program's goal "to set a man on the moon" (Kitano et al. 1997). Kitano and Asada argue that, although it might seem overambitious, the goal has to be set high enough to ensure that technical breakthroughs are necessary to reach the goal. Although landing on the moon did not have a direct economic impact, the technologies developed during the process "formed the powerful technological, and human foundations of American industry" (Kitano & Asada 1998). Beating the soccer World Cup winner with a team of robots could provide technologies that will similarly affect industry and society. For example, soccer provides a domain where robots must learn to understand other robots and collaborate to play the game in a dynamic real-time environment successfully. Robots that possess these skills could be deployed in real-life scenarios where they have to collaborate with humans and robots. In addition, soccer has rules for how the players are allowed to touch opponents physically. Complying to these rules requires the robots to learn how to not hurt humans in situations where physical touch is necessary (Kitano & Asada 1998).
The RoboCup competitions are generally split into two research directions: real robot leagues, where physical robots play soccer and the software (simulator) robot league, where simulated agents play soccer via a centralised Soccer Server (Kitano & Asada 1998). In contrast to the physical robots league, the RoboCup Soccer Simulation League 2D (SS2D), where the agents are represented as two-dimensional

objects, is focused on the research of high-level behaviours since researchers and developers do not have to deal with the problem that the agents can not physically execute the behaviours. Therefore, SS2D can be used to develop and evaluate high-level behaviours and integrate the algorithms later in real robots when they are ready (Budden et al. 2015).

The most common offensive behaviours performed by a player possessing the ball in (SS2D) soccer are passing, dribbling and shooting. The successful execution of these behaviours makes it possible to advance the ball towards the opponents' goal, which increases the chance of winning the game. In particular, an intelligent pass strategy can lead to dominating the game by prolonged ball possession, and better stamina management (Zare et al. 2022). Furthermore, an effective dribble can be the optimal solution for situations where, for example, a defender blocks the path by moving the ball to a better position with more free space (Zare et al. 2021). As a consequence, effective dribbling can result in more opportunities to shoot on goal and more successful key passes, which increases the probability of scoring goals (Leal et al. 2021). Finally, effective shooting increases the scoring chance by targeting the shot at the goal where defenders and the goalkeeper can not block it.
Predicting the passing, shooting and dribbling behaviour of the own teams' agents provides several benefits to improve the teams' and agents' gameplay. First, pass behaviour prediction enhances the teams' strategy and the agents' pass accuracy, decision-making and stamina management (Zare et al. 2022). Second, shot prediction might similarly improve the agents' shot accuracy and decision-making. Third, dribble prediction could enhance the teams' strategy and agents' decision-making and stamina management.

1.2 Previous work

Zare et al. (2022) modelled pass behaviour by predicting the intended receiver of a pass with a Deep Neural Network (DNN) (and Random Forest). To achieve this, Zare et al. built a "Data Extractor" module into the agents to generate the training data for the models by recording and producing features of the ball and players of both teams (the agents' observations) during the game. In addition, engineered features (such as determining the nearest and riskiest opponents) are produced from the agents' observations. Zare et al. also showed the importance of these engineered features for the pass target player predictions. The labels are produced by recording the corresponding decision-making of the agents, which is based on their observations.

This online recording of observations combined with the agents' decision-making had never been explored for agents' behaviour prediction in SS2D, according to Zare et al. (2022). The training data used in all previous approaches for modelling behaviour was obtained from the games' log files. Log files are game records (generated after the game has finished) and contain information such as the position of the ball and players. However, log files do not give access to the agents' actual observation and corresponding decision-making process during the game (Zare et al. 2022). Furthermore, most previous methods only relied on features of positions of the ball and players during the game, while the effect of other features is not considered.

1.3 Research question

Zare et al. predicted the target player of a pass with 84% accuracy, which validated the effectiveness of using the Data Extractor module compared to previous methods that relied on log files (Zare et al. 2022). Furthermore, the pass target player classification provides the team's agents benefits during the game, such as increased passing accuracy, as described earlier. However, predicting the teammates' shooting and dribbling behaviours separately to validate the Data Extractor module has (to my knowledge) not been reported. These behaviours have an intended target location on the soccer field instead of a target player, making them regression problems instead of classification. Furthermore, the pass's target location can also be predicted separately since the pass location is not always near the target player's location. Finally, the importance of the (engineered) features for modelling the shooting and dribbling behaviours can be investigated.
Consequently, this thesis examines the following research question: "How accurate can deep neural networks predict the agents' passing, shooting and dribbling decisions made in a SS2D match by shifting from classification to regression when the models are trained with (engineered) features generated by the Data Extractor module?"

Zare et al. (2022) reported that the pass target player prediction accuracy of 84% was an improvement up to 5% compared to previous methods. The data for this model was generated by playing against the same team in *full-state mode*, which ensures that the agents receive the exact observations. It is difficult to compare the performance of a classification model to a regression model directly. Therefore, an improvement of 5% in performance is expected instead for the regression models compared to previous methods when the same data is generated to train these models. Furthermore, Zare et al. showed the importance of the engineered features for the pass target player prediction, which is also expected for the dribbling and

shooting predictions.

1.4 Outline

The theory section provides the necessary knowledge to understand the data generated by the Data Extractor, consisting of the soccer agents' observations and corresponding decision-making in the SS2D environment. In addition, the theory's last section gives insight into how the agents can benefit from the DNN's predictions. Next, the first part of the research method section describes the general method of generating the data, designing the regression models, and determining the importance of features for these models. The second part of the method describes the used datasets and models per behaviour in more detail. The results and evaluation section describes the performance of the models per behaviour combined with a feature importance analysis. In addition, the impact of filtering unimportant features on the models' performance is measured. Finally, the performance of the best model per behaviour is analysed in more detail.

Chapter 2

Theory

The "2D Soccer Server" section first describes the SS2D soccer field to understand the coordinates of the target locations for the behaviours. Second, the communication between the server and the agents is described to understand the agents' observations (the training features) sent by the server and the agents' response to the server (the target labels). Next, the HELIOS agent2D sample team section describes the characteristics of the agents of the used team for generating the data. This is achieved by describing its components, the teams' formation, corresponding player roles and the decision-making process of the Agent2D players. The Data Extractor module section describes the components of the module, how it generates data by simulating matches, and the relevant features and labels of the data are described. Finally, the deep neural networks section describes the motivation for using them by describing the benefits of predicting the agents' behaviours and its place in the decision-making framework.

2.1 2D Soccer Server

Two teams of eleven autonomous players can play a match of soccer in the dynamic and real-time environment of the 2D Soccer Server Kitano & Asada (1998). The soccer field and all objects in the 2D soccer simulation environment are described by a 2-dimensional (x,y)-coordinate system that uses the same dimensions of real soccer fields defined by the FIFA soccer rules. Soccer fields are 105 meters in length (which corresponds to the x-coordinates) and 68 meters in width (which corresponds to the y-coordinates). The goal's width is the only deviation from the official rules; it is two times the original width: 14.02 meters. This width was necessary because scoring goals was too difficult considering the two dimensions of the environment (and consequently, the goal does not have height) (Noda 1995). The movable objects on the SS2D field are the players and the ball, while the static

visible objects are the goals, flags and field lines (Noda 1995). Flags are integrated into the environment to assist the agents' localisation of the important parts of the field. The flags are visible as small white circles in figure 2.1. For example, the centre of the field is the origin and is described by the flag *(f c)*. Therefore, the positive x-coordinates entail the right side of the middle line (x = 0 to x = 52.5) and the left part of the field ranges from x = 0 to x = -52.5. Similarly, the top of the field is described by positive y-values (y = 0 to y = 34), and the bottom of the field by negative y-values (y = 0 to y = -34) (Chen et al. 2001). Therefore, player 9 of the yellow team is located at (x,y) = (18, -25) while player 5 is located at (x,y) = (-2, 15) in figure 2.1. The rest of this thesis assumes that the left-playing yellow team is the team with data extractor, which implicates that the coordinates are never mirrored (the opponents' goal is always located at x = 52.5 and never at x = -52,5.)

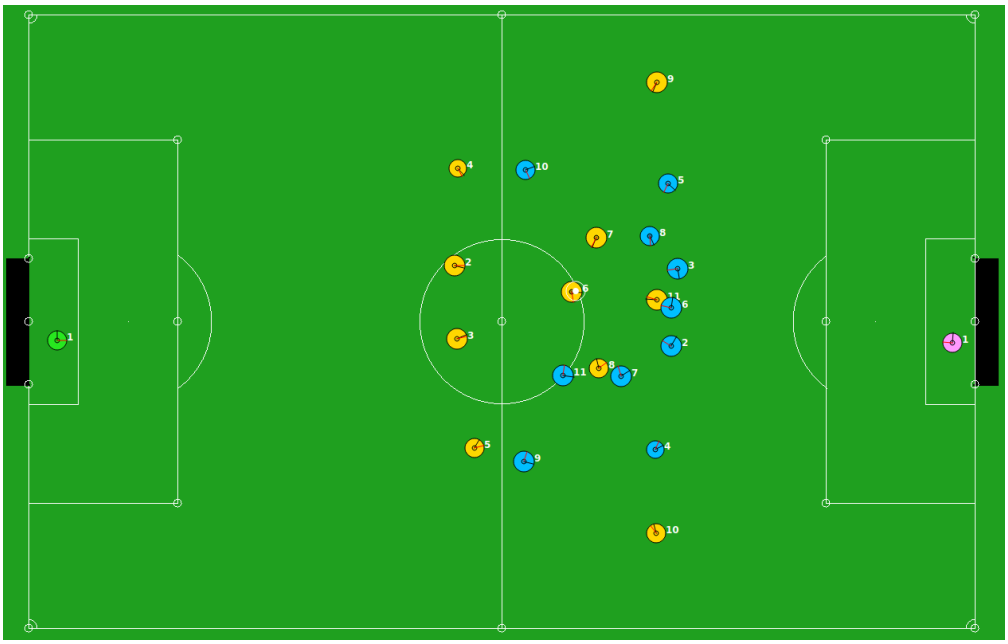


Figure 2.1: A visualisation of the 2D soccer field. Player 9 is located at (x,y) = (18,-25).

The sent information to the agents by the server, the agents' possible responses to this information, and how the server handles the agents' responses are detailed below.
The soccer servers' client sensor protocol determines which information is sent to each playing soccer agent. Agents have three sensor models to receive different kinds of information; the aural sensor model, visual sensor model and body sensor

model. These models combined give the agents an accurate impression of the environment. The aural sensor model allows the agents to "hear" messages from the referee, coach or other players while the body sensor detects the "physical status" such as the kicking power of the player (Chen et al. 2001). The visual sensor provides the agents visual information of objects (i.e. the ball, players, lines and flags) on the field that the player currently sees. The server sends information to the agents at the start of every cycle based on the players' view angle in the previous cycle. A match is simulated in 6000 cycles of 0.1 seconds, so the server sends information to the agents 6000 times per match.

The server sends the visual information in a standard format with the object's name, distance, direction and relative change in position compared to the previous cycle. Therefore, the visual information differs per player since players are located at different positions and looking in different directions. Agents usually have a limited view distance, resulting in more information loss at further distances. For example, for a certain distance (depending on settings and the agents' abilities), players can no longer identify an agent's uniform number. For even further distances, the players can not identify to which team a particular player belongs (Chen et al. 2001). However, the server can also send the exact observations to the agents when fullstate mode is turned on.

The agents use their action models to send commands to the server to act based on the sensed information about the environment. These action models describe the possible low-level behaviours that the agents can perform, which includes a dash model, kick model and turn (neck) model. Dashing accelerates an agent in the direction of its body. The server only executes a player's kick command when the player is not offside and close enough to the ball (the player has to be within the "kickable margin", which differs per player). Finally, the turn command allows a player to change its body direction, and the turn_neck command changes the neck direction relative to the body's direction. Although an agent can execute low-level behaviours such as dashing, kicking and turning, methods for executing high-level behaviours such as performing a dribble which requires a combination of dashing, kicking and turning had to be developed. Therefore, the agent2D starter team was released by HELIOS to provide new researchers with a sample team with players that can execute basic (team) strategies by performing the high-level soccer behaviours such as dribbling, passing and shooting (Akiyama & Nakashima 2013).

2.2 Agent2D sample team

The agent2D sample team, one of the components of the open-source HELIOS base code, allows beginning researchers or competing teams to directly simulate a competitive 2D soccer match (Akiyama & Nakashima 2013). The *librcsc″*, another component, handles the necessary communication between the server and the agents. Twenty per cent of the participating teams in the RoboCup competition of 2007 used HELIOS as the base code. This percentage increased yearly until it became the most popular base code, with 83% of teams using it, in 2012. The top-5 teams of the 2021 (CYRUS, HELIOS2021, YuShan2021, HfutEngine2021 and Alice2021) will all use the HELIOS agent2D team as base team for the RoboCup 2022 competition.

Two essential components of a competitive SS2D soccer team are (1) the team formation, which regulates the positioning of the individual players and (2) the team's strategy to successfully execute cooperative behaviours amongst the teammates. The "formation framework" and "online multi-agent planning framework" are integrated into the agent2D team to effectively handle the two necessary components for the agent2D sample team.

2.2.1 Formation framework

The formation framework, player roles and corresponding positioning of the agent2D sample team players are relevant for interpreting the differences between players in the target locations of their passing, shooting and dribbling behaviour. For example, the left forward player executes dribbles on the left side of the field instead of on the right side of the field. Figure 2.2 illustrates the used formation of the agent2D sample team (the yellow players). The roles and positioning of the players are pre-defined in formation configuration files. The players can be recognised by their *uniform number* (a unique number for each player) such that the player 1 is the goalkeeper. Players 2 and 3 are the defensive centre backs, while players 4 and 5 are the left defensive back and right defensive back players respectively. The midfield consists of players 6 (the ball holder in figure 2.2), 7 and 8, which are the defensive midfielder, left attacking midfielder and right attacking midfielder, respectively. Finally, Player 9 is the left forward, player 10 the right forward and player 11 plays at the striker (centre) position. The players generally keep this formation, but the individual positioning changes based on the ball's position, which is also defined in the formation files (Akiyama & Nakashima 2013).

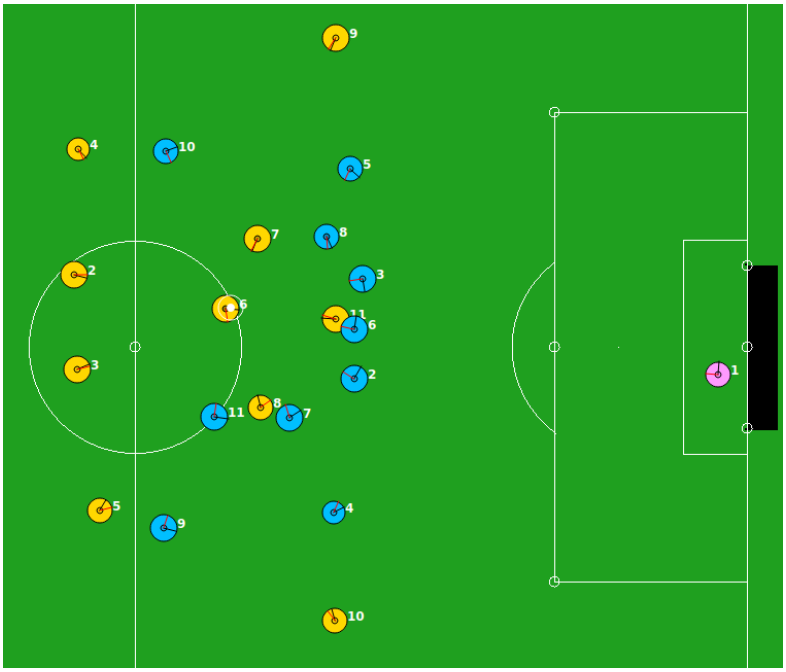


Figure 2.2: Illustration of the formation of the used agent2D sample team (yellow) where the players can be recognised by their unique uniform number.

2.2.2 Decision-making framework

The decision-making framework of the agent2D players gives insight to the produced target labels generated by the Data Extractor module. A soccer team performs optimally when the teams' players can effectively execute *cooperative behaviour* (Akiyama et al. 2012). In the planning framework, cooperative behaviour is defined as a chain of kick actions (pass, dribble or shoot) amongst multiple teammates to achieve the best possible terminal state (i.e. reach the opponents' goal to score). This optimal action sequence has to be planned amongst the teammates, which requires generating multiple action plans in the form of action trees. Consequently, the optimal decision of the current ball holder is the action that eventually leads to the highest evaluated chain action, which requires an effective and flexible tree search framework.
Therefore, the "online tree search framework" is integrated into the agent2D sample team to produce cooperative behaviour (Akiyama et al. 2012). Kick actions are defined as nodes in a tree where each path from the root node (the ball holders' current situation) to a terminal node represents one of the possible action plans. The framework contains three modules to generate and evaluate the kick action chains: the *CooperativeAction* module, *ActionGenerator* module and *Evaluator* module.

The Cooperative action module defines the kick actions (represented as nodes in the tree) as an object with the following relevant attributes: the type of action, the x-target and y-target position of the action, the uniform number of the ball holder (that performs the action) and receiver (only relevant in case of a pass). The ActionGenerator module creates CooperativeAction objects based on the input of the environment and the predicted outcome of the action. If the outcome of the action is predicted to be a success, then the action is stored as a candidate combined with its predicted outcome as an *ActionStatePair*.

Four categories of the CooperativeAction objects exist: hold, pass, dribble and shoot. However, the ActionGenerator generates candidates for sub-categories of the dribble and pass, while for shoot only one category exists. The following ActionGenerators exist, which generate a different number of candidates with different methods.

Hold The agent holds the ball without moving

1. No patterns are generated since it is a static action. Therefore, the hold action is also not considered further in this thesis.

Pass The agent passes the ball to another player

1. Direct pass: ten candidate locations for this pass type are generated near the receiver's position.
2. Lead pass: 2250 candidate locations, where the half of the candidates is sampled at the receiver's position, while the other half is targeted towards the opponents' goal.
3. Through pass: Generates candidates targeted towards the opponents' goal behind the defense.
4. Cross: generates 300 candidates in the penalty area around the position of the intended receiver.

Dribble The agent moves with the ball towards a target location.

1. Short Dribble: Generates 48 candidates of short distance dribbles.
2. Long dribble (self-pass): Generates 300 candidates of long distance dribbles.
3. Queued dribble: Agents moves while it is searching the action tree for the optimal game plan.

Shoot The agent shoots at the opponents' goal.

1. Generates 25 candidates uniformly sampled at the goal of the opponent.

The Evaluator module evaluates the action chains consisting of generated the actions as described above by giving a real value to each action in the chain. The value of the individual actions in the chain are calculated by, for example, considering positions of the ball, ball holder and opponents. In addition, actions that are not blocked when executed are evaluated higher. Consequently, the action at the root node that starts the action chain with the highest total value is chosen as action by the ball holder in the current cycle.

2.3 Data Extractor module

The Data Extractor module consists of two main components built into the agent2D base team; the *Feature Extractor* and the *Label Generator*, as shown in figure 2.3. The feature extractor records and engineers features of the agents' current observation at the start of the cycle. These features are only recorded for the agent in possession of the ball and if the agent plays in the "Data Extractor team". In this thesis, the fullstate mode is turned on which results in exact observations for the agents. This ensures that the results are reproducible and the difference between models is not caused by noise differences in the used datasets. However, Zare et al. (2021) showed that the performance of the models decreases when fullstate mode is off. The label generator records the ball holder's decisions, generated by the *Chain Action* module at the end of a cycle.

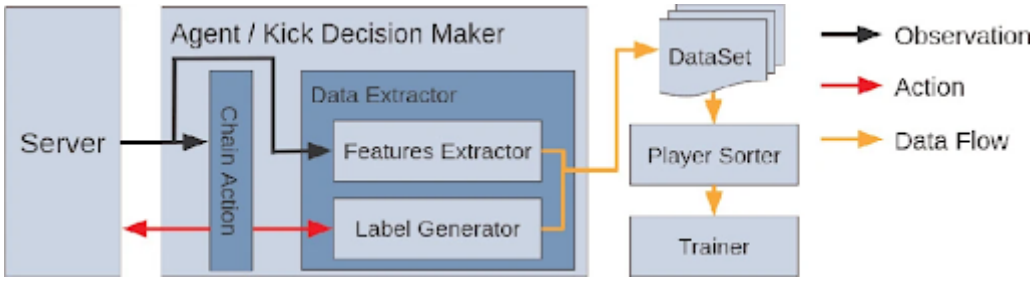


Figure 2.3: The implementation of the Data Extractor module into the agent2D decision-making framework. Figure is extracted from (Zare et al. 2022)

The most relevant features generated by the Data Extractor can be divided into the feature groups described below Zare et al. (2022). The Data Extractor provides the option to record a feature group for all objects on the field (the players of both teams and the ball), only for the own teams' players or only for the kicker of the own team. The "position and velocity" group and "relative to kicker" group are measured for all objects. The "player features" are recorded for all players. The "top 2-riskiest opponents", "top 2-nearest opponents" and "distance to the

goal" are measured for the players of the own team. Finally, the "dribble angles" are only measured for the kicker. This results in 928 features described in the "dataset information" section in the appendix.

Position and velocity The position and velocity features, measured in both Cartesian (x,y)-coordinates and Polar (R,T)-coordinates.

Relative to kicker Position and velocity features measured relative to the kicker (the ball holder). These relative positions and velocities make the situation translation-invariant; similar situations can be identified on different parts of the field Zare et al. (2022).

Player features The unique abilities of each player such as its kicking power. In addition, the agents' team side and unique number are recorded.

Top 2-riskiest opponents calculates the two riskiest opponents for intercepting a pass.

Top 2-nearest opponents describes the two nearest opponents to each player

Dribble angles measures the distance to the nearest opponent in 12 angle ranges (such as between 30 and 60 degrees) for the kicker

Distance to the goal describes players' distance and angle towards the opponents' goal

The labels recorded by the label generator originate from the Chain action module, which produces an agents' decision as a cooperative action object (as described in the agent2D sample team section). Therefore, the optimal action, target player's uniform number (only for a pass), target location (x,y), ball speed and action angle are recorded as labels. In addition, the description of the action generated by the Actiongenerator module is recorded.
The 928 features are sorted in the following order: ball, own teams' players and the opponents' players. The players of the teams can be sorted in four different ways: (the "field evaluator"-sorting mentioned in (Zare et al. 2022) is removed from the Data Extractor).

1. Uniform sorting: The default sorting method where agents are sorted by their uniform number. The highest uniform numbers are put before the lower uniform numbers in the order of features.

2. X-sorting: sorts the features of the players based on their x-position on the field.

3. Kicker-First: pushes the features of the kicker (of the own team) towards the front of the dataset (after the ball features).

Therefore, the following combinations exist: "uniform sorting", "uniform sorting, kicker first", "x-sorting" and "x-sorting, kicker first). An index number (the agents' position in the sorted data) is therefore necessary to identify the players besides their uniform number since the position of the players in the data changes.

2.4 Deep neural networks

The dataset obtained by the Data Extractor module can be used to train a deep neural network (DNN) for predicting the agents' soccer behaviour. A DNN is suitable for predicting the varied characteristics of the behaviours because it finds complex patterns in the dataset by forming "multiple layers of abstraction" of the input data (LeCun et al. 2015). The accurate prediction of the behaviours benefits the players during a match. In general, the predictions can benefit the (next) ball holder, the team strategy and teammates' stamina management and anticipation can be improved (Zare et al. 2022).

The (next) ball holder can use the DNN prediction to increase its action accuracy, better decision-making and possibly anticipate better on the current situation. The server sends information about the environment to the agents at the beginning of every cycle (as mentioned in the soccer server subsection). When a team is in possession at the beginning of the cycle, the ball could be either moving towards the next ball holder or owned by the current ball holder (or moving towards the goal by a shot of the last ball holder). The upcoming ball holder could use the server's information as input for the trained DNN to predict its behaviour before receiving the ball.
The accuracy of a pass (and possibly also of a shot targeted at the opponents' goal) can be improved with this prediction by using it as input for an algorithm that updates the neck angle to optimise the agents' view towards the target player or location (Zare et al. 2022). This provides the agent extra time to execute its behaviour accurately. Furthermore, the decision-making could be enhanced with the prediction since it allows the agent to search the chain action tree deeper.

The team strategy and stamina-management can be improved by communicating the prediction to the other players so they can anticipate the following action of the ball holder earlier. For example, the DNN predicts that the action will be a pass towards player 8. Player 8 can use this information as input to its "unmarking algorithm", which calculates where the receiving target player should run

towards to lose its marking defender ("unmarking"). Furthermore, if player 4 knows that player 8 will likely receive the ball instead of itself, then player 4 does not have to move towards the target or execute its "unmarking" behaviour, which costs stamina. Therefore, the limited stamina of player 4 can be used for different situations instead (Zare et al. 2022). Similar algorithms for anticipating the dribbles of teammates could be implemented which use the dribble prediction as input.

Python-based deep learning libraries such as Keras are popular because the models are easy to implement. However, a Keras model can not be directly implemented in the agent2D framework since the computation time makes it impracticable considering the 100ms of a cycle. In contrast, all agent2D source code is written in c++ because it is faster than Python. Therefore, the CYRUS team released the CppDNN, which uses the stored weights of a trained Keras model to recreate it (Zare et al. 2021).

To summarise the complete method: the Data Extractor module collects features of the environment and produces engineered features labelled by the characteristics of the resulting behaviour. The Keras model is trained with the dataset, and its weights are stored in the CppDNN. This library recreates the Keras DNN, while maintaining a fast computation time. The Feature Extractor method of the Data Extractor serves the CppDNN the input to predict the following behaviour. This prediction is used to provide the benefits such as updating the neck angle as described above. The updated neck angle is sent back to the server to receive the optimised visual information in the next cycle. Furthermore, the prediction can be communicated to the teammates to use as input for the "unmarking algorithm".

Chapter 3

Research method

The research method contains the following steps: first, data has to be generated to be able to train the models. Next, the general training script has to be produced. The third step includes optimising the models for each behaviour. The fourth step is to determine the feature importance for each model. The final step is to evaluate the models and optimise the results by filtering out features based on the feature importance analysis.

The "Data generation" section describes how the data is generated and which datasets are obtained. The "General training script" section describes the shared components of the different models. The "pass prediction", "dribble prediction" and "shot prediction" sections describe the target variables and models for predicting the corresponding behaviours. The "Hyperparameter tuning" section describes how the base model is optimised with the Keras tuner, and the "feature importance" section describes the method for determining the importance of individual features. The "evaluation" section describes the method of evaluating the classification and regression models by considering the evaluation measures and their analysis.

3.1 Data generation

The data is generated by simulating matches between the Agent2D team with the build-in Data Extractor module (the "Data Extractor" team) and the agent2D team (version 3.1.1) without the module. The simulation of one match would take 10 minutes to execute manually. Zare et al. (2022) simulated 3000 matches to generate the dataset, which would manually take around 21 days.

However, "sync-mode" can be turned on in the simulator, which reduces the simulation time of a match from ten minutes to one minute. Furthermore, Cyrus'

Autotest2D [1] provides a script to simulate matches in parallel by using multiple threads. The datasets are created by running 200 rounds with five threads to create the data of 1000 matches. The combination of the Autotest2D and "sync mode" option decreases simulation time from 7 days to 4 hours for the simulation of 1000 matches.

Table 3.1 shows the performance of the Data Extractor team (left team in the table) against the Agent2D team (right team) in simulations 1 and 2. Simulation 3 shows the performance against CYRUS (right team), which shows that the agent2D team can not compete with CYRUS. The randomness of simulating soccer games causes a slight difference between the simulations against the Agent2D team. However, the difference between individual values of the two simulations is at most 0.13. The simulations are executed in "full state" mode to minimise the effect of random noise on the performance, making it possible to reproduce the results and compare the models with different datasets as input.

	Simulation 1	Simulation 2	Simulation 3
Number of games	1000	1000	250
Average goals (left : right)	3.94 : 4.27	3.95 : 4.19	0.56 : 11.62
Average goal difference	-0.33	-0.24	-11.06
Average points (left : right)	1.25 : 1.62	1.31 : 1.55	0.0 : 3.0
average point difference	-0.37	-0.24	-3.0

Table 3.1: Independent simulation 1 and simulation 2 show similar results when the Data Extractor team plays against Agent2D. Simulation 3 shows that CYRUS outperforms the Data Extractor team. The simulations are performed in full state mode.

Furthermore, table 3.2 shows that the performed actions are not evenly distributed since 72.5% of the behaviours is a dribble, and 22.5% is passing behaviour. Shots are only 3.8% of the performed behaviours in simulation 1. This difference requires different sizes of training data per behaviour. The pass data of 3000 matches equals the dribbling data of 1000 matches, while the shooting data of 6000 matches would be necessary to achieve this. 18 simulations for shooting data (to equal the size of pass data) would not be feasible for every sorting method. Therefore, the shooting data of 4000 matches is used, equalling 100 thousand shot instances. The simulation of 1000 matches produces more than 10 thousand data files because a separate file is created for every player in every match. Therefore, these files are merged into one data file, which equals around 6.5 gigabytes.

[1] `https://github.com/Cyrus2D/AutoTest2D`

Performed behaviours in different simulations

	Simulation 1	Simulation 2	Simulation 3
Total behaviours of team:	674551	671819	102086
Ball hold:	8430 (1.2%)	8640 (1.3%)	9714 (9.5%)
Dribbles:	488811 (72.5%)	487523 (72.3%)	44532 (43.6%)
Passes:	151940 (22.5%)	150405 (22.3%)	47247 (46.3%)
Shots on goal:	25370 (3.8%)	25249 (3.7%)	587 (0.57%)

Table 3.2: Number of behaviours per simulation of a 1000 matches. Simulation 1 and 2 show similar performed behaviour patterns Simulation 3 shows less dribbling behaviour and in proportion more passes

Extra labels are added to the Data Extractor module to make the visualisation and evaluation more convenient. First, the original labels are added to the data next to the scaled labels. Furthermore, the uniform number of the kicker is added since the existing "is_kicker" feature is a binary feature ("1" if a player is the kicker and "0" otherwise) which requires more processing to identify the kicker in every row. Finally, the recording of the dribble type is added to the Data Extractor module, which was not available before. Three categories are identified by running a simulation that outputs the strings of the dribble types. Three types of dribbles were identified with this method: the long dribble (self-pass), short dribble and queued dribble.

3.2 General training script

The models for predicting the characteristics of the different behaviours have similarities and differences in structure and design. The models' shared components are described in this section and the differences per behaviour in the corresponding sections. Tensorflow version 2.9.1. is used, and the model training is done on the NVidia Titan V (12GB) GPU.

The script file used for predicting the pass target player of Zare et al. (2022) is used as a template for the general training script and for designing the models per behaviour. The model in the template is (partly) implemented as a Keras sequential model. The sequential API is convenient for single output models since it has a straightforward design. However, the sequential API is not designed for handling multiple outputs, which is necessary for all behaviour models except the shot prediction model. Therefore, the Keras functional API is used because it can handle multiple outputs (and inputs).
The functional API allows to put in an array of output layers, with each layer

having its loss function. The combined loss of the model is the sum of the outputs losses, which can be used to select the best performing model across the multiple outputs by saving the model with the minimal combined validation loss. The mean squared error is used as a loss function for all regression outputs since it "punishes" larger errors, and categorical cross-entropy is used as the loss function for all classification tasks. The pass classification and regression tasks are handled with separate models since the MSE loss of the regression outputs are larger than the categorical cross-entropy loss of a classification output. Therefore, it would be necessary to balance the losses, which takes more effort than keeping the models separate.

The network used by Zare et al. (2022) for the pass target classification (defined as the "base model") contains five dense layers with 1024, 526, 256, 64 and 32 units in the layers respectively. A drop-out layer was implemented after the first dense layers. In addition, this model has one output layer with 11 units (for each receiving players) and contains a softmax function for the player classification. The models for the regression tasks are designed by replacing the original output layer. The pass regression model has two output layers (the x-target and y-target) without softmax. The dribble model has three output layers (x-target, y-target and distance), while the shot model contains one output layer for the y-target without softmax. Furthermore, the layers with 256, 64 and 32 units (the final three hidden layers) are optimised for each model, as described in the section below. Furthermore, the drop-out layer is optimised and the optimal learning rate for the optimiser is determined.

3.3 Hyperparameter tuning

The best combination of the number of units in the three deepest layers, dropout rate and learning rate of the base model (described above) are determined with the Keras tuner (O'Malley et al. 2019). The units of the third dense layer are varied between 256 units and 512 units with steps of 64 units to consider whether more units increase the performance. The fourth and fifth dense layers are tested with 32, 64, 128 and 256 units. Finally, dropout rates of 0.1, 0.2 and 0.3 (after the first dense layer) and learning rates of 0.01, 0.001 and 0.0001 are tested. The number of epochs is retrieved by configuring the optimal combination and setting a maximum of 125 epochs to find the epoch in this range with the minimal validation loss.
The Hyperband tuner is chosen out of the four available Keras tuners because it finds the optimal configuration fast. To achieve this, the Hyperband method exploits random search in combination with "adaptive resource allocation and early-stopping" (Li et al. 2017). The adaptive resource allocation depends on the *successive halving algorithm* which considers the performance of all configurations

in every round and takes the top half of best-performing configurations to the next round until one configuration is left. This process is sped up in the hyperband tuner by training a small number of epochs for the models in the first rounds and increasing this number when fewer models are left. Furthermore, the early-stopping stops a model's training when the model's performance has not increased for a chosen number of epochs (O'Malley et al. 2019).

3.4 Feature importance

The *permutation importance* method of the Eli5 library[2] can determine the importance of features for a model's predictions, which helps to explain the predictions of machine learning models. The "permutation importance" method determines the importance of a feature by measuring how much the performance of a model declines (or increases when it is confusing) by removing the feature. First, the model is trained regularly on all features. Second, one feature is removed from the test set in each step to measure the impact on the trained models' performance. The feature is not actually removed since a model expects the same input dimensions. Therefore, the values of a feature become random noise, which could let the trained model fail when this noise is not drawn from the feature's original distribution. Therefore, the values of a feature are permuted, and hence the method's name is "permutation importance".

The *feature importances* attribute contains the mean decrease (or increase) of the models' performance for each feature after fitting the permutation importance object on the test data. Therefore, the features with the highest positive importance are the most important features for the trained model since these features have the most significant effect on performance when removed. Features with zero importance have no impact on the prediction performance. Negative feature importance implies that the score increases when the feature is removed from the test set, which signals that these features confuse the trained model. Therefore, the effect of performance after filtering these features can be tested. The relative importance of the features is calculated by dividing the feature importance of a individual feature with the sum of the features' importances combined.

To summarise the method for modelling each behaviour: the architecture of the model is determined with the Hyperband tuner. Next, the models are trained with the different sorting methods to compare the performance against the baseline. The penultimate step is to determine the importance of features and filter the confusing and redundant features from the training dataset. The final step is to retrain the model with the feature-filtered dataset and test its performance.

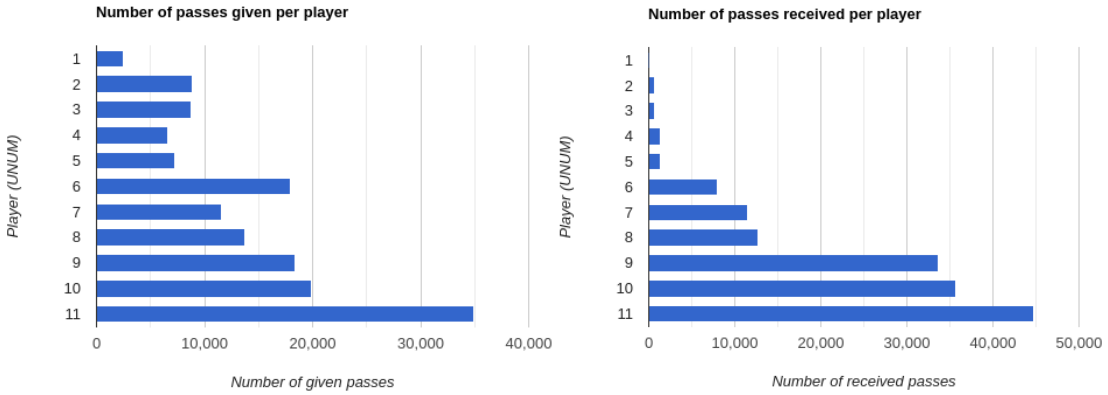
[2] `https://eli5.readthedocs.io/en/latest/`

3.5 Pass prediction

The pass dataset section describes the characteristics of the past behaviours, and the "pass prediction models" section describes the used models for predicting these characteristics.

3.5.1 Pass dataset

Figure 3.1 shows that the number of sent and received passes per player is not evenly distributed; the attackers (players 9, 10 and 11) receive the most passes while the goalkeeper and defenders (players 1, 2, 3, 4 and 5) receive few passes. Players 6 and 11 send the most passes. Table 3.4 shows that the average distance between the location of the receiver and the pass target location is 4.37. Therefore, it is relevant to not only predict the target player of a pass but also the target position coordinates of a pass.



(a) Sent passes per player

(b) Received passes per player

Figure 3.1: Received and sent passes per player

Distribution type of pass

	Pass dataset 1
Total passes	151683
Lead pass	114603 (75.6%)
Through pass	23799 (15.7%)
Cross	11431 (7.5%)
Direct pass	1850 (1.2%)

Table 3.3: Distribution of type of passes

Table 3.4 shows the average absolute x difference, absolute y difference and total (Euclidean) distance between the pass target player's location and the pass's target location. The data in table 3.4 shows that the direct pass is the nearest pass to the target player with an average distance of 0.32, while the through ball has the largest average distance of 11.03. The agents' different intentions per pass type (described in the decision-framework section) are confirmed by these distances.

	x-difference (abs)	y-difference (abs)	Total distance
Passes combined	3.52	1.89	4.37
Lead pass	2.40	1.78	3.35
Through pass	10.28	3.18	11.03
Cross	1.20	0.63	1.42
Direct pass	0.22	0.20	0.32

Table 3.4: Average absolute x difference, y-difference and total euclidean distance of target player position compared to pass target position

3.5.2 Pass prediction models

This section describes the models for predicting the pass target player, pass type and pass target location. The first step is to (attempt to) reproduce the described results described in (Zare et al. 2022). This reproduction includes predicting the target player's uniform number with the different sorting methods as input for the "pass classification base model". Next, the "optimised pass classification model" is used to predict the target player and pass type. Finally, the pass regression model predicts the pass target location (x,y). The Keras tuner determines the architecture and learning rates of the models.

Pass classification base model The "base model" of Zare et al. as described in the general training script section. A learning rate of 0.001 is used for the baseline and a learning rate of 0.0001 is used for traing with all features.

Optimised pass classification model The final three dense layers have 320, 128 and 128 units respectively, and the dropout rate after the first dense layer is 0.3. The optimal learning rate is 0.0001

Pass regression model The final three dense layers have 448, 32, 32 units respectively, and the optimal dropout is 0.2. The best learning rate is 0.0001.

3.6 Shot target predictions

3.6.1 Shooting dataset

The simulation of 4000 matches produces around 100 thousand instances of shooting data, which averages 25 shots per match. The distribution of shots taken per player in figure 3.2 shows that the forward players 9, 10 and 11 shoot 94.3% of the shots. The attacking midfielders 7 and 8 only take 6.7% of the shots. The shots taken by players 1 to 6 are not included since these players took less than 0.1% of the shots.

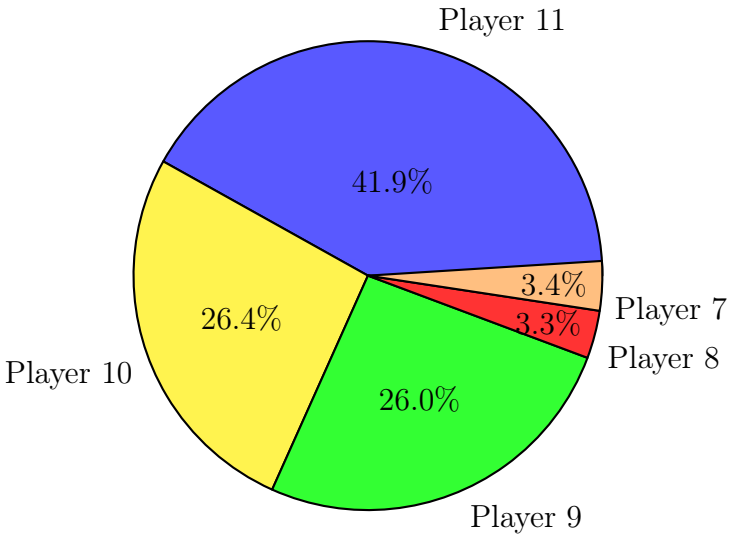


Figure 3.2: Distribution of number of shots per player. Players 1 to 6 are not included in the figure

The y-target of shots the players takes is always in the range of the goal (y = -7.01 to y = 7.01). The middle of the goal is defined as y-target = 0. Therefore, positive y-targets are the right side of the goal, and negative y-targets correspond to the left side of the goal. Finally, They-target is the only relevant label for the shot prediction models since the x-target of the shots is always set as the goal-line (x = 52.5).
The attacking players have different "favourite" shot targets. Figure 3.3 shows the distribution of the shots of all players combined and the attacking players' shots separately. Most shots are taken either near the left or right for the players combined. However, the distributions differ per player. Player 11 (the striker) has a varied shooting behaviour pattern with three favourite targets: the middle of the

left side of the goal and the left and the right post. Player 10 shoots almost always at the right post, while player 9 does the opposite by shooting at the left post. Player 8 (the right attacking midfielder) also has the right post as the favourite target but targets the goal with more variation compared to player 10. Finally, player 7 (the left attacking midfielder) has a similar pattern as player 8 but is mirrored.

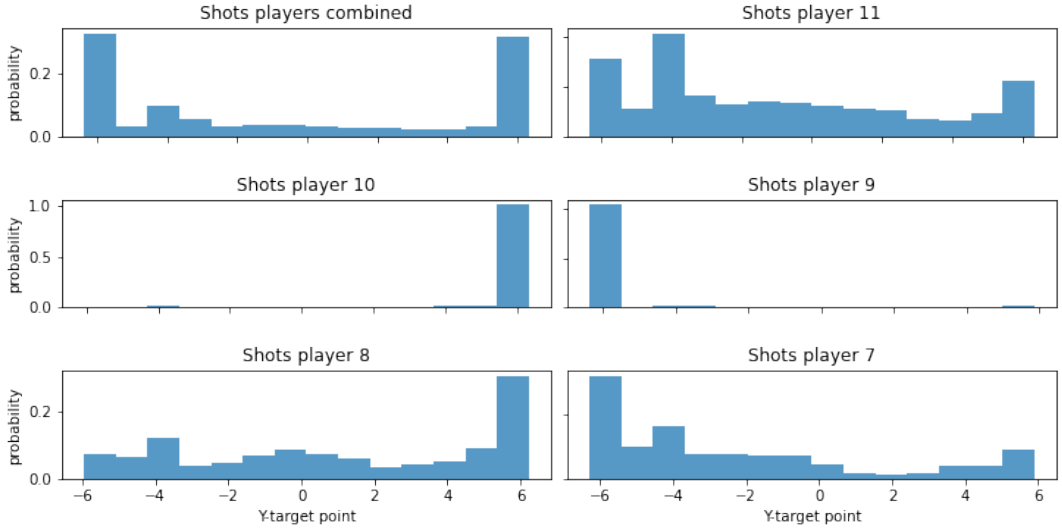


Figure 3.3: Probability of shooting target of all players combined (upper left) and per attacking player

3.6.2 Shot prediction models

Two models are tested and compared to achieve the lowest mean absolute error in predicting the target of the shots on goal. The "feature filtered shot model" is designed by considering the agents' shot target decision-making. The decision depends on information related to the kicker itself, the defenders and the goalkeeper. Therefore, including the teammates' features is redundant for the model. A downside of this approach is that this can only be tested with data sorted with the kicker first attribute. Without this attribute, the features of the kicker would be present in different columns for each shooting instance. This would require selecting different columns per training instance, which is impossible when training the model. The first two dense layers of both models have 1024 and 525 units equal to the "base model".

Shot model Optimised "base model" of Zare et al. where the final three dense layers have 448, 32 and 256 units respectively, and the optimiser has a learning rate of 0.0001. The optimal learning rate for the baseline is 0.001.

Feature filtered shot model The features of the teammates are filtered from the model. The final three layers have 384, 128 and 32 units as optimal configurations. The optimisers' learning rate is 0.0001.

3.7 Dribble prediction

3.7.1 Dribble dataset

Simulating 1000 matches produces about 500 thousand dribble instances. The existing relevant labels for predicting dribbling behaviour are the x-target and y-target of the dribble. In addition, the dribble distance is created as an extra-label since the starting point is not considered in predicting the target location. This distance is defined as the Euclidean distance between the ball's position at the start of the dribble and the target location. The average dribble distance for all dribbles combined is 3.34 meters, long dribbles ("self-passes") are the longest dribbles with an average distance of 5.4 meters, short dribbles cover a median distance of 2.4 meters while queued dribbles have an average distance of 1.11 meters 0.6 meters.

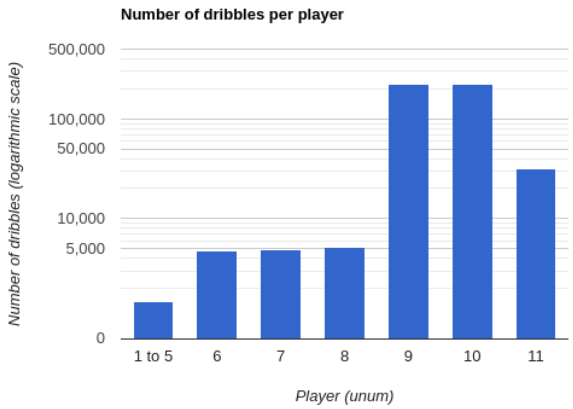


Figure 3.4: Number of dribbles on logarithmic scale

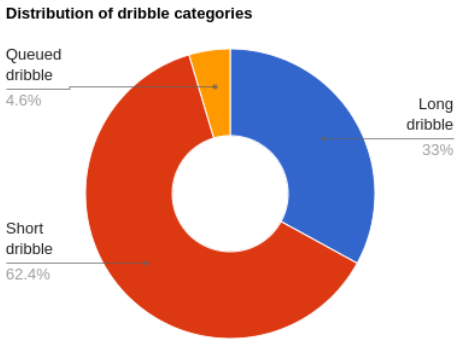


Figure 3.5: Number of dribbles per category

3.7.2 Dribble prediction model

The dribble model has three outputs to predict the x-target, y-target and dribble distance of the agents' dribble decisions.

Dribble model The first two dense layers have 1024 and 526 units, respectively (equal to the "base model"). The final three dense layers of the optimised model have 320, 256 and 32 units and a learning rate of 0.0001 is used for the optimiser. A learning rate of 0.001 is used for the baseline.

3.8 Evaluation method

3.8.1 Evaluation measures

The classification models (for predicting the pass target player and pass type) are evaluated with accuracy, the proportion of correctly predicted labels. The regression models (for predicting the x-target, y-target and distance) are evaluated with the mean absolute error (MAE) and (adjusted) R^2-score. The MAE is used to evaluate the average performance of the model instead of the root mean squared error (RMSE) because it is unambiguous, and the average error is described more naturally (Willmott & Matsuura 2005). The R^2-score measures the proportion of the target labels' variance that the features can explain in a range of 0 to 1. The maximum score of 1 is achieved when the features can explain all variance in the dependent variable. The R^2-score has the advantage that the performance on the different regression tasks (with different value ranges) can be compared because the measure is independent of the domain (Chicco et al. 2021).

In contrast, the MAE can be any value in the range of 0 to infinity, depending on the target variables' value range. Therefore, comparing the MAE of two target variables by the same model is inappropriate. For example, comparing the MAE of the x-target with the MAE of the distance is not meaningful since these variables have different value ranges. However, the variance in evaluating the R^2-scores is dataset-dependent. Therefore, the adjusted R^2-score can be used, which takes the test sample size and number of features into account.

3.8.2 Analysis

The evaluation measures described above can be used to summarise the models' overall performance. However, the models perform differently under different circumstances. For example, the shot model has a higher prediction error for shots of midfielders compared to shots of defenders. Therefore, it is useful also to analyse the performance of the best models per behaviour across situations.

The situations are distinguished by saving the true labels, the predictions of the best performing model, the action description (e.g. "short dribble) and the uniform number of the kicker. As a result, the labels and predictions can be filtered to only measure the models' performance for the different action types and per kicking (or receiving in case of passing) player.

Chapter 4

Results and evaluation

The pass, shoot and dribble prediction results are described in the following order: first, the model's baseline and performance with all features as input are compared. Next, the effect of filtering out features with negative importance or selecting only the positive features is shown if it improved the performance. Next, The predictions of the best-performing model are visualised and analysed. The last section per behaviour contains an analysis of the feature importance (which was used for the feature selection).
The baseline is defined by Zare et al. (2022) as using only the positional (x, y) features of the ball and the agents (i.e. 46 features). In addition, the used dataset for the baseline is sorted uniform without 'kicker first'.

4.1 Pass prediction results

The "pass target player and type" section contains the attempt to reproduce the 84% accuracy in predicting the uniform number of the pass target player (classification) of Zare et al. (2022). In addition, the results of predicting the pass type are described. The "pass target location" section describes the results obtained with the pass regression model by predicting the pass's (x,y)-target location.

4.1.1 Pass target player and type (classification)

Table 4.1 shows the performance of the optimised pass classification model in predicting the pass target player. The 84% accuracy in previous work was achieved with the uniform sorting method, which is 2% higher than the 82% accuracy achieved in this reproduction attempt. Furthermore, the accuracy is 1% higher than the baseline (instead of 5%).
In addition, table 4.1 shows that the optimised classification base model predicts

the correct pass type with 89% accuracy when the model is trained with uniform sorting and the kicker first, which is again 1% higher than the baselines' performance. The performance did not improve by selecting features based on their feature importance. Therefore, these results can be found in table 7.3 in the "pass prediction results" section in the appendix.

Method	Player Accuracy	Type accuracy	Average Accuracy
Baseline	0.81	0.88	0.845
Uniform	**0.82**	0.88	0.85
Uniform, kicker	**0.82**	**0.89**	**0.855**
X	0.81	0.88	0.845
X, kicker	0.81	0.88	0.845

Table 4.1: Pass target player and pass type accuracy with different models

The accuracy per receiving target player is shown in table 4.2. The best model predicts the correct player with 90% accuracy on average when the correct player is an attacker. On the other hand, predicting the defenders or midfielders as being the receiver is at least 20% lower than the overall accuracy of 82%. However, these players receive fewer passes than the attackers.

Player	Accuracy	Passes received
11	0.90	20313
10	0.89	16391
9	0.91	15115
8	0.61	5654
7	0.58	5143
6	0.58	3616
5	0.33	614
4	0.39	649
3	0.26	304
2	0.37	321

Table 4.2: Accuracy per receiving player

Table 4.3 shows that the pass type being a lead pass is correctly predicted in 95% of the cases. The model can not predict the direct pass (but the direct pass only entails 1.2% of the passes). Furthermore, the type of pass being a cross is

predicted better than the through pass, while the number of through passes given is more than two times the number of crosses.

Pass type	Accuracy	Number of passes
Direct pass	0.03	868
Lead pass	0.95	51510
Through pass	0.72	10768
Cross	0.84	4974

Table 4.3: Accuracy per type of pass

4.1.2 Pass target location (regression)

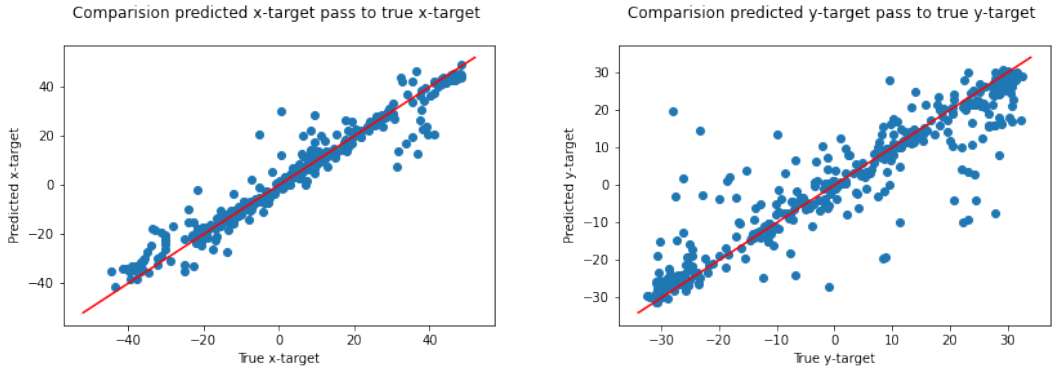
The MAE and adjusted R^2-scores for the x-target and y-target predictions are shown in table 4.4. The MAE scores are minimal for the uniform sorting method, which decreased by 0.29 for the x-target and 0.41 for the y-target compared to the baseline. However, the adjusted R^2-scores are equal to the score of the baseline. In addition, the performance did not improve by filtering out features (as determined by the feature importance method described in the next section). Therefore, table 7.4 and table 7.5 can be found "pass prediction results" section in the appendix.

Method	x-target MAE	y-target MAE	x-target adjusted R2-score	y-target adjusted R2-score
Baseline	3.18	5.32	0.95	0.81
Uniform	**2.89**	**4.91**	0.95	0.81
Uniform, kicker	3.05	5.12	0.95	0.81
X	3.15	5.33	0.95	0.81
X, kicker	3.19	5.55	0.95	0.81

Table 4.4: MAE scores and average adjusted r2-score per method

Table 4.4 also shows that the R^2-score of the y-target predictions is 14% lower than the x-target for all methods. This performance difference is also visible in figure 4.1, which shows the predictions of the uniform sorted model (in blue dots) compared to the true target values (the red line) for both the x-target and y-target. 400 random samples are plotted instead of all predictions to ensure that the blue dots (predictions) can still be distinguished. The MAE, standard deviation and median absolute error are 2.98, 3.88 and 1.71, respectively for the x-targets' error

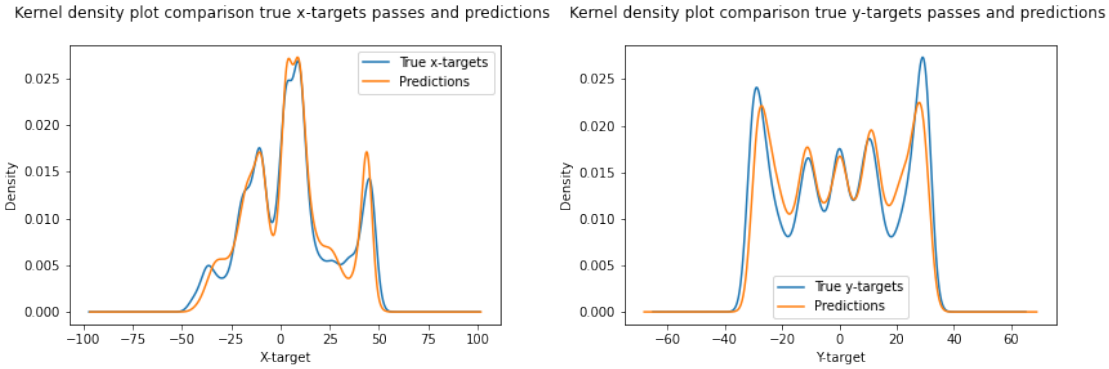
distribution, while the same measures are 4.91, 7.03 and 2.34 for the y-targets' error distribution.



(a) X MAE: 2.98, standard deviation: 3.88 (b) Y MAE: 4.91, standard deviation: 7.04

Figure 4.1: Sample of 400 samples compared to predictions

Finally, the kernel density estimations (KDE's) of the x-target and y-target labels are compared to the corresponding predictions of the uniform sorted model in figure 4.2. The figure shows that the x-target predictions KDE better fits the original distribution than the y-target predictions.



(a) X target KDE (b) Y-target KDE

Figure 4.2: KDE comparison

4.1.3 Pass feature importance

The feature importance for predicting the pass target player (classification) can be found in table 7.6 and table 7.7 since this is already reported in (Zare et al. 2022). In addition, the feature importance for the pass type is also reported in table 7.8

and table 7.9 because it produced similar results to the pass target player feature importance.

615 out of the 927 have positive importance for predicting the x-target, 154 have zero impact, and 159 have negative importance. The y-target prediction has less positive importance features (523) and more negative importance features (251). Table 4.5 shows that the offside count, which measures the accuracy of the offside line is important for the x-target prediction. Table 4.6 shows that the features related to determining the riskiest opponents has the most negative importance for the x-target prediction. The most positive importance and negative features for the y-target can be found in table 7.10 and 7.11 in the "pass prediction results" in the appendix.

feature name	Relative Importance (%)
offside_count	22.74
p_l_2_kicker_x	4.96
p_l_2_pass_opp2_dist_diffbody	3.06
ball_pos_x	1.66
p_l_2_pos_r	1.55
p_l_2_pos_x	1.50
p_l_11_kicker_x	1.49
p_l_2_pass_opp2_angle	1.28
p_l_1_pos_x	1.26
p_l_2_angle_goal_center_r	1.17
p_l_2_pass_opp1_dist_proj	1.14
p_l_10_kicker_x	1.11
p_l_3_pass_opp1_dist_proj	1.09
p_l_3_pass_opp1_dist_line	0.91
p_l_9_kicker_x	0.90
p_l_2_pass_opp1_dist	0.88
p_l_4_pass_opp1_dist_proj	0.88
p_l_3_pass_opp2_dist_diffbody	0.83
p_l_8_kicker_x	0.79
p_r_11_kicker_x	0.78

Table 4.5: Twenty most important features for pass x-target prediction

feature name	Relative Importance (%)
p_l_10_pass_opp2_dist_proj	-14.19
p_l_11_pass_opp2_dist_proj	-10.00
p_l_11_pass_opp2_dist_line	-9.70
p_l_6_pass_opp2_dist	-5.26
p_l_6_pass_opp2_dist_proj	-3.74
p_l_11_pass_opp1_dist	-3.70
p_l_9_pass_opp1_dist	-3.08
p_r_6_kicker_t	-2.54
p_l_4_face	-2.52
p_r_6_kicker_x	-2.34
dribble_angle_2	-2.09
p_r_3_vel_r	-2.09
p_r_1_pos_t	-2.07
p_l_7_pass_opp2_dist_proj	-1.96
p_l_6_kicker_t	-1.92
p_r_6_face	-1.92
p_l_5_pass_opp2_dist_line	-1.79
p_l_11_kicker_t	-1.67
p_r_8_pos_r	-1.45
p_l_9_pass_opp1_angle	-1.30

Table 4.6: Twenty most negative features for pass x-target prediction

4.2 Shot prediction results

Table 4.7 describes the shot model's best-obtained performance for the baseline and all features as input with the four different sorting methods. The R^2-scores of uniform sorting and uniform sorting with kicker-first kicker are both 91%, which is 5% higher than the baseline. Furthermore, the MAE of 0.69 is 0.34 lower than the baseline.

Sorting method	MAE	R^2-score	Adjusted R^2-score
Baseline	1.03	0.86	0.86
Uniform	**0.69**	**0.91**	**0.91**
Uniform, kicker	**0.69**	**0.91**	**0.91**
X	0.71	0.90	0.89
X, kicker	0.70	0.90	0.90

Table 4.7: Shot model: MAE and (adjusted) R^2-score of baseline and all features with the four sorting methods

The first and second row of table 4.8 show that the feature filtered shot model, where the features of the teammates are filtered, performed 2% better on adjusted R^2-score compared to using all features (93% for uniform sorting and 92% for x-sorting). The third and fourth row show that the MAE can be decreased by 0.03 by filtering the features with negative importance (based on analysis described in the next section). In total, 93% of the variance can be explained, and the MAE is 0.58 for the best performing feature filtered shot model.

Sorting method	Input features	MAE	R^2	Adjusted R^2
Uniform, kicker	Kicker, opponents	0.61	0.93	0.93
X, kicker	Kicker, opponents	0.64	0.92	0.92
Uniform, kicker	Positive feature importance	0.59	0.93	0.93
Uniform, kicker	Non-negative feature importance	**0.58**	0.93	0.93

Table 4.8: (Feature) filtered shot model: MAE and (adjusted) R^2- score

However, the minimal MAE of 0.59 is 0.5 higher than the median absolute error of 0.093. In addition, the 95th percentile is an MAE of 2.5, the 99th percentile is an MAE of 6.33, and the maximum error is 12. Figure 4.3 illustrates this skewed absolute error distribution with its long tail of rare large errors.

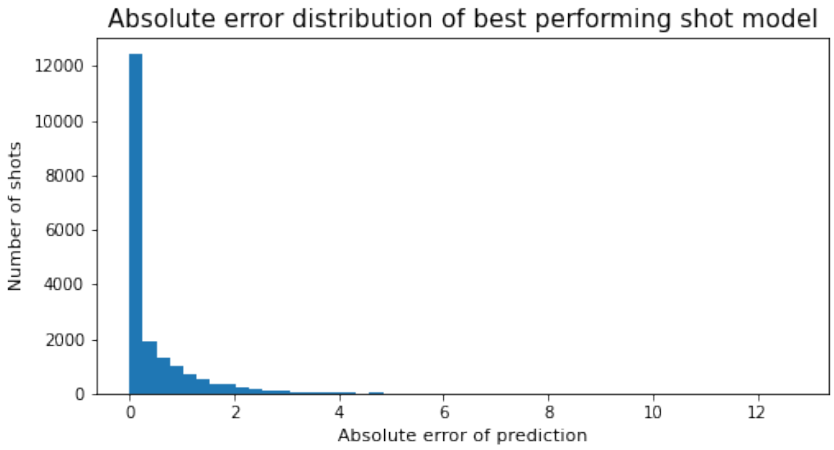


Figure 4.3: The absolute error distribution of the best performing model with a MAE of 0.59 and a median error of 0.093. The distribution has a long tail

Table 4.9 shows ten random test samples with the best model's predictions, absolute error and the kicker to illustrate the model's performance. A larger random sample of size 45 can be found in table 7.12 in the "shot prediction results" section in the appendix.

Y-target	Prediction	Absolute error	Kicker
-0.229	0.208	0.437	11.0
-6.109	-6.159	0.05	11.0
-6.019	-6.031	0.011	9.0
-3.691	-3.72	0.029	11.0
6.132	6.177	0.045	11.0
3.602	2.757	0.845	11.0
6.151	6.177	0.027	10.0
1.248	2.026	0.777	11.0
-5.51	-3.632	1.878	10.0
5.993	6.029	0.036	10.0

Table 4.9: Random sample of 10 predictions of the best performing shot model

The kernel density estimate comparison in figure 4.4 shows that the best model's predictions and actual shot target label distribution are similar. The density of the true shots is slightly higher around the peaks at y-target = -6 and y-target = 6 (left goal post and right goal post), while the models' predictions have too much density near y-target = -4. A similar plot for x-sorting can be found in

figure 7.2 in the "shot prediction results" section in the appendix.

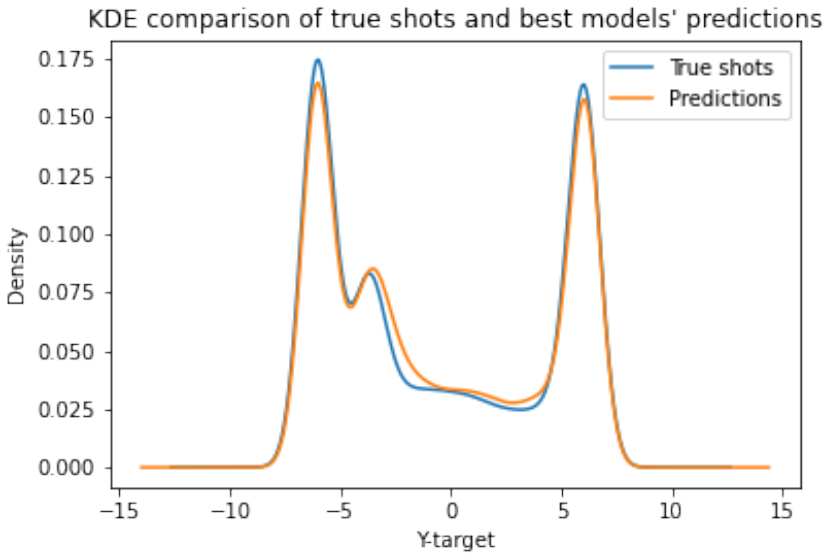


Figure 4.4: Kernel density estimate comparison of best model's predictions compared the true targets of the shots.

The prediction performance of the attackers and attacking midfielders are shown in figure 4.10. The predictions of shots of player 9 and player 10 have the lowest MAE with 0.27 and 0.2 respectively. The MAE of player 11 is almost three times higher than that of player 10, while the R^2-scores are similar. The shots of player 7 and player 8 are the most difficult to predict.

Player	Shots	MAE	R^2	Adjusted R^2
11	8407	0.94	0.822	0.817
10	5226	0.27	0.831	0.824
9	5174	0.20	0.883	**0.877**
8	605	1.09	0.744	0.410
7	662	1.19	0.636	0.549

Table 4.10: Shot prediction performance per attacking player

4.2.1 Shooting feature importance

472 of the 928 features have positive feature importance, 94 have zero feature importance, and 274 have negative feature importance for predicting the shot

target. Table 4.11 shows that the features relating to the goalkeeper (p_r_11) are the most important features for the model's predictions of the shot target. Furthermore, 6 features belong to the kicker group, which measures the relative positions. Finally, the dribble_angles 4, 5 and 6 are in the top twenty, which calculates the distance to the nearest opponent in each angle range between -60 and 60 degrees with steps of 30 degrees. In total, the top 20 features are responsible for 76.6% of the importance.

feature_name	Relative Importance (%)
p_r_11_kicker_y	29.64
p_r_11_body	24.48
p_r_11_kicker_t	5.61
p_l_3_pass_opp2_dist_diffbody	2.90
p_r_11_kicker_x	2.06
p_r_11_vel_t	1.85
p_l_1_kicking	1.24
p_r_11_vel_r	1.19
dribble_angle_4	1.00
ball_kicker_t	0.95
p_r_11_face	0.82
dribble_angle_6	0.69
p_l_11_kicker_y	0.69
dribble_angle_7	0.64
p_r_10_kicker_x	0.62
p_r_10_kicker_y	0.54
p_r_11_pos_y	0.45
p_l_11_body	0.44
dribble_angle_5	0.41
p_r_9_vel_t	0.40

Table 4.11: Twenty most important features for shooting

Table 4.12 shows that the features related to determining the riskiest opponents for intercepting a pass (pass_opp1 features) decrease the performance of the model the most.

feature_name	Relative Importance (%)
p_l_3_body	-4.20
p_r_5_vel_t	-3.19
p_l_1_pass_opp1_angle	-3.04
p_l_2_pass_opp2_dist_diffbody	-2.99
p_l_4_pass_opp2_dist_line	-2.86
p_l_1_pass_opp1_dist_diffbody	-2.53
p_l_4_pass_opp2_dist	-2.17
p_l_1_pass_opp1_dist_line	-2.06
p_r_6_vel_t	-2.01
p_l_1_pass_opp1_dist_proj	-1.87
p_l_3_pass_opp2_dist_line	-1.68
p_r_5_vel_r	-1.68
p_l_2_pass_opp2_dist_line	-1.54
p_l_1_pass_opp1_dist	-1.51
p_l_5_vel_t	-1.41
p_r_8_tackling	-1.37
p_l_4_pass_opp2_dist_proj	-1.31
p_l_8_near1_opp_angle	-1.20
p_l_4_face	-1.20
p_l_7_vel_t	-1.18

Table 4.12: Twenty most negative features for shooting

4.3 Dribble prediction results

This section first describes the baseline's performance and the dribble model's performance with all features as input. Next, the effect of filtering out features (based on feature importances analysed in the last subsection) is described. The performance of the best model (uniform sorted and "kicker first" with positive importance features for either the x-target or y-target) is visualised and analysed. The final section includes the analysis of the feature importance for the dribble prediction.
738 out of the 97479 (0.75%) predictions are filtered since these predictions had a small negative predicted distance. Furthermore, the "queued dribbles" are not evaluated (4.8% of the dribbles) for the performance of the model since the median distance of these dribbles is only 0.6 meters.

Table 4.13 shows that the dribble model reached the best performance when the input features were sorted uniform and with "kicker first". The average MAE of 0.79 for the predicted targets combined decreased 0.13 compared to the baseline. Furthermore, the average adjusted R^2-score is 3% higher. The x-sorting method reached a 90% R^2-score, which is 2% lower than the other sorting methods.

Method	x-target MAE	y-target MAE	Distance MAE	Average MAE	Average adjusted R^2-score
Baseline	0.91	0.99	0.87	0.92	0.89
Uniform	0.79	0.92	0.74	0.82	0.92
Uniform, kicker	0.81	0.85	0.72	**0.79**	0.92
X	0.84	0.99	0.79	0.87	0.90
X, kicker	0.83	0.86	0.73	0.81	0.92

Table 4.13: Dribble model's performance for the baseline and all features with the four sorting methods. "Uniform, kicker" reached the best performance.

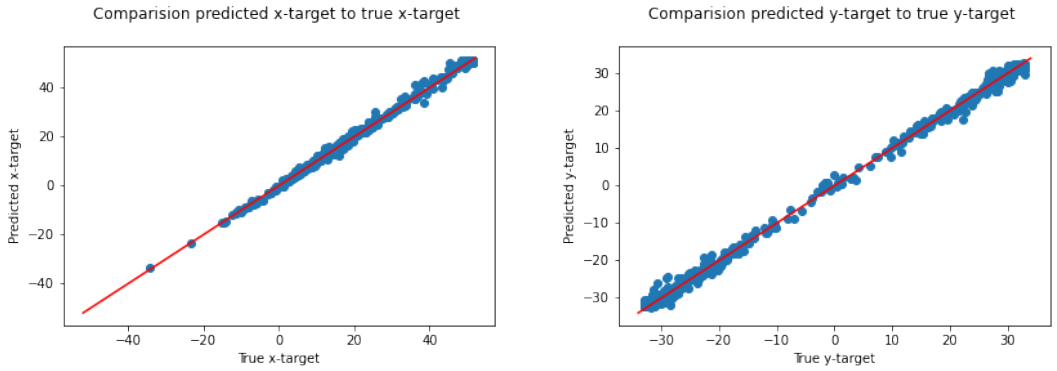
Table 4.14 shows that selecting the features with positive importance for either the x-target or the y-target ($x > 0$ or $y > 0$) provides the lowest average MAE of 0.70. This decreases the average MAE from 0.79 to 0.70 compared to the best performance with using all features. In addition, table 4.15 shows that this feature importance filter, where 707 out of the 928 features are used, improves the average adjusted R^2-score from 92% to 94%. In total, the average R^2-score is 5% higher compared to the baseline and the average MAE decreased with 0.29. This dribble model's predictions are further analysed in remaining tables and figures of this section.

Feature importance filter	Features	x-target MAE	y-target MAE	Distance MAE	Average MAE
$x > 0$ and $y > 0$	505	0.79	0.89	0.70	0.79
$x \geq 0$ and $y \geq 0$	680	0.80	0.89	0.68	0.79
$x > 0$ or $y > 0$	707	**0.67**	**0.80**	**0.63**	**0.70**
$x \geq 0$ or $y \geq 0$	882	0.84	0.92	0.77	0.84

Table 4.14: Performance of the model with four different combinations of filtering features with negative importance or the selection of positive importance features

Feature importance filter	Features	R^2-score: x-target, y-target, distance	Adjusted R^2: x-target, y-target, distance	Average R^2-score uniform	Average Adjusted R^2-score uniform
$x > 0$ and $y > 0$	505	0.997, 0.998, 0.787	0.997, 0.998, 0.786	0.927	0.927
$x \geq 0$ and $y \geq 0$	680	0.997, 0.998, 0.799	0.997, 0.998, 0.798	0.931	0.931
$x > 0$ or $y > 0$	707	0.998, 0.998, **0.826**	0.998, 0.998, **0.824**	**0.941**	**0.940**
$x \geq 0$ or $y \geq 0$	882	0.997, 0.998, 0.73	0.997, 0.998, 0.723	0.906	0.906

Table 4.15: R^2-scores of best model's predictions with different combinations of features filtered.



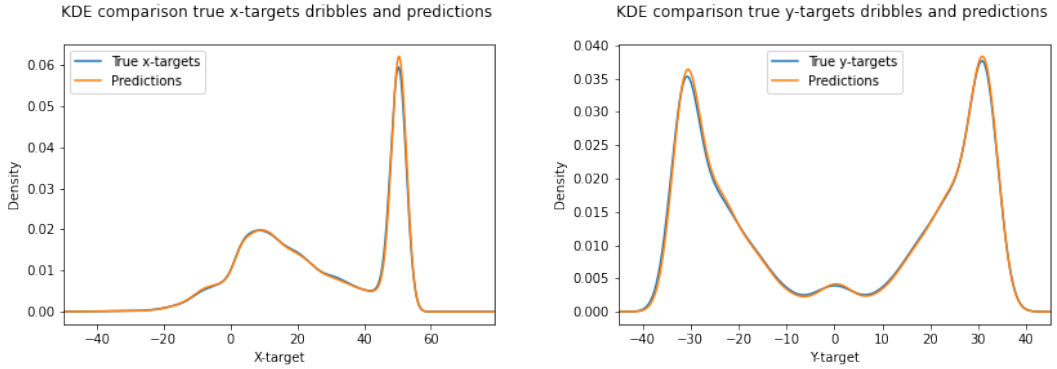
(a) x-target sample predictions

(b) y-target sample predictions

Figure 4.5: Random sample of 500 predictions (blue dots) compared to true target values (red line)

Figure 4.5 shows a random sample of 500 predictions (as blue dots) compared to the true target values for both the x-target and y-target. 500 random samples are used for visualisation purposes to ensure that the predictions can be distinguished.

The y-target predictions have a larger MAE and standard deviation (0.80 and 0.81) compared to the x-target predictions (0.66 and 0.72). Figure 4.6 shows that the distributions of the target labels and the corresponding predictions are nearly identical. Table 4.16 shows 10 random samples of the predictions. A larger sample of size 40 can be found in table 7.13 in the "dribble prediction results" section in the appendix.



(a) X-target KDE comparison

(b) Y-target KDE comparison

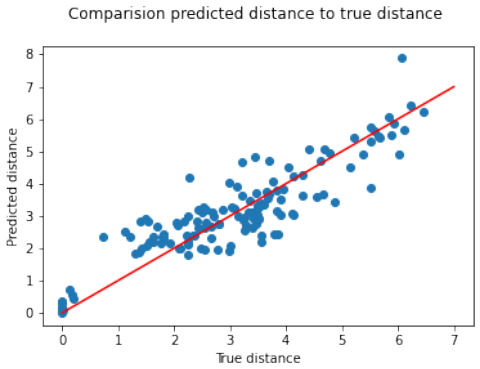
Figure 4.6: Kernel density estimation (KDE) comparison of the true dribble target labels and the predicted dribble target labels

x target	x prediction	y target	y prediction	Distance	Distance prediction
19.332	18.923	-32.308	-32.197	3.689	2.8
51.103	51.089	18.717	19.177	3.15	3.021
51.219	50.514	-25.633	-24.12	2.586	3.392
50.778	51.291	-17.333	-17.496	2.329	2.228
8.936	8.427	-30.276	-30.031	0.027	0.048
5.218	5.592	19.831	22.212	2.869	3.684
51.169	50.629	24.88	23.929	2.938	2.916
50.487	51.224	-18.813	-18.285	2.621	2.663
23.903	23.169	30.861	31.961	3.634	3.165
12.762	12.147	-31.337	-31.073	3.317	2.778

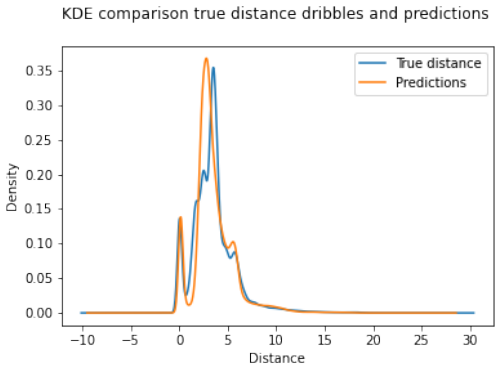
Table 4.16: 10 random samples of the predictions of the best performing dribble model

The R^2-score of the distance prediction of the best dribble model is 82.4%, which is 17% lower compared to the x-target and y-target R^2-scores. This is also visible in figure 4.7. Figure 4.7a shows 150 samples of the true distance compared

to the predicted distance. The 150 samples and scaling to maximum distance of 7 (the 95th percentile) are used for visualisation. The MAE is 0.63 while its standard deviation is 0.70. In addition, figure 4.7b shows that the kernel density estimate of the predictions and the true distance labels have differences.



(a) 150 samples of true distance compared to predicted distance, scaled to maximum distance of 7 (95th percentile) for visualisation



(b) KDE comparison distances

Figure 4.7: True distance compared to the best model's dribble predictions

The dribbles of player 9 and player 10 are predicted the most accurate with R^2-scores of 93%, as shown in table 3.5. Only the performance for the attackers and midfielders is shown since the defenders executed not enough dribbles. Furthermore, the dribbles of the striker (player 11) are also predicted accurately with an average adjusted R^2-score of 0.89. Finally, the performance of the midfielders is more difficult to predict with average adjusted R^2-scores of 0.74, 0.72 and 0.70 for player 8, player 7 and player 6 respectively.

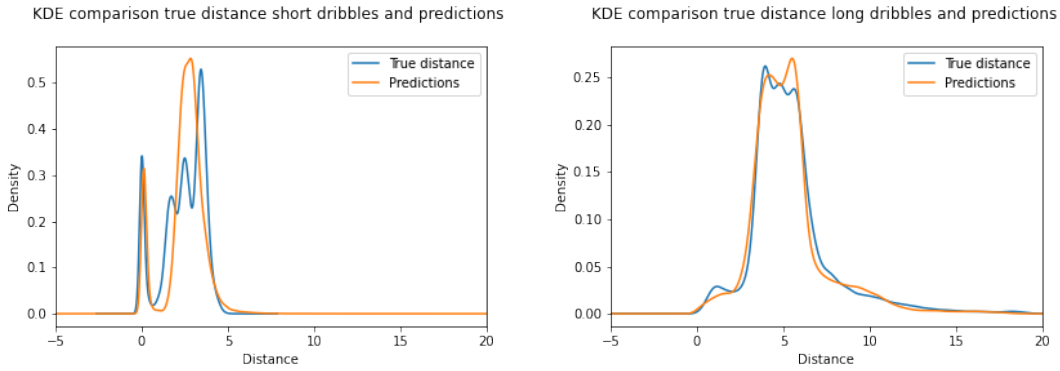
Player	x-target MAE	y-target MAE	Distance MAE	Average MAE	Average adjusted R^2-score
11	0.80	1.22	0.71	0.91	0.89
10	0.66	0.76	0.63	0.69	**0.93**
9	0.65	0.76	0.61	0.68	**0.93**
8	0.67	0.94	0.55	0.72	0.74
7	0.61	0.93	0.61	0.72	0.72
6	0.54	1.02	0.49	0.68	0.70

Table 4.17: Performance of dribble performance per player

Finally, table 4.18 shows that short dribbles are more difficult to predict than long dribbles (self-passes). The average adjusted R^2-score of the long dribble is 6% higher. This performance difference is also visible when comparing the kernel density estimations for the short dribble and long dribble (both compared to their corresponding true labels), as shown in figure 4.8.

Category	x-target MAE	y-target MAE	Distance MAE	Average MAE	Average adjusted R^2-score
Short dribble	0.63	0.80	0.57	0.67	0.86
Long dribble	0.73	0.81	0.73	0.76	**0.92**

Table 4.18: The performance per type of dribble



(a) Short dribbles

(b) Long dribbles

Figure 4.8: KDE comparison true and predicted (long and short dribble)

4.3.1 Dribble feature importance

Determining the importance of the features for predicting the dribble behaviour is done separately for the x-target and y-target of the dribble. 611 out of the 928 features have positive importance for determining the x-target of dribbles, 175 features have zero impact (the redundant features) on the performance and 142 features negatively impact the predicting the x-target (the confusing features). The number of important features has a similar distribution for predicting the y-target: 601 features have positive importance, 175 features have no impact and 152 features have a negative impact. This leads to the following combined number of positive or non-negative features for predicting the x-target and y-target together:

1. 505 features have positive impact on both variables
2. 680 features have non-negative impact on both variables
3. 707 features have a positive impact on either one of the variables
4. 882 features have a non-negative impact on either one of the variables

feature name	Relative Importance (%)
p_r_11_kicker_x	6.93
p_l_6_pass_opp2_dist_line	4.01
p_r_10_kicker_x	2.63
p_r_9_kicker_x	2.49
ball_pos_x	2.02
p_r_11_kicker_r	1.98
p_l_6_pass_opp2_dist	1.96
p_r_7_kicker_x	1.95
p_l_1_pos_x	1.78
p_l_11_kicker_x	1.77
p_r_8_kicker_x	1.76
p_l_3_angle_goal_center_r	1.57
p_l_2_angle_goal_center_r	1.52
p_l_1_angle_goal_center_r	1.50
p_r_7_kicker_y	1.35
p_r_8_kicker_y	1.25
p_l_4_angle_goal_center_r	1.13
p_l_9_kicker_x	1.10
p_l_8_kicker_x	1.10
p_l_5_angle_goal_center_r	1.10

Table 4.19: twenty most important features for prediction of x-target dribble

Table 4.19 shows the twenty features with the most relative importance for predicting the x-target of a dribble. These twenty features combined have a relative importance of 40%. Consequently, the 591 other positive importance features equal 60% of the importance. The figure shows that the features belonging to the x-positions of objects and the "kicker" features (the relative positions to the kicker) are important for the x-target dribble predictions. Furthermore, 5 of the 20 most

important features are the angle_goal_center_r, which measures the distance of the player to the origin of the goal.

feature name	Relative Importance (%)
p_r_1_kicker_t	-6.64
p_l_6_kicker_t	-6.30
p_l_1_angle_goal_center_t	-4.08
p_l_7_pos_y	-3.86
p_l_3_kicker_t	-3.55
p_r_10_kicker_t	-3.43
p_l_6_vel_r	-3.06
p_l_9_pos_y	-2.73
p_l_3_face	-2.61
dribble_angle_1	-2.61
p_l_11_pos_t	-2.60
p_l_6_angle_goal_center_t	-2.57
p_r_11_kicker_t	-2.49
p_l_7_body	-2.41
dribble_angle_8	-2.34
p_l_6_face	-2.33
p_l_2_angle_goal_center_t	-2.23
p_l_3_open_goal_angle	-2.05
p_l_3_body	-2.00
p_r_1_pos_t	-1.96

Table 4.20: Twenty most negative features for predicting the x-target of the dribble

On the other hand, table 4.20 shows the twenty features with the most negative importance for predicting the x-target. These twenty features account for 60% of the features that decrease the models' performance. The other 122 features with negative importance account for the other 40%. The features ending with "_t", which indicate the polar coordinate theta are decreasing the performance of the model the most. The polar coordinate theta measures the polar angle.

Feature name	Relative Importance (%)
p_r_8_kicker_y	9.60
p_r_7_kicker_y	8.28
ball_pos_y	6.54
p_l_3_kicker_y	6.35
p_r_11_kicker_y	5.56
p_l_1_pos_y	5.44
p_r_9_kicker_y	5.27
p_r_10_kicker_y	4.18
p_l_8_kicker_y	3.30
p_l_10_kicker_y	2.75
p_l_11_kicker_y	2.73
p_r_2_kicker_y	2.46
p_r_1_kicker_y	1.67
p_l_7_kicker_y	1.59
p_r_4_kicker_y	1.53
p_l_3_pos_y	1.46
p_l_9_kicker_y	1.29
p_l_10_kicker_t	1.22
p_l_5_kicker_y	1.11
p_l_5_pass_opp2 _dist_line	1.06

Table 4.21: Twenty most important features for predicting the y-target of the dribble

Table 4.21 shows that the top twenty most important features for predicting the y-target of a dribble has similar patterns as predicting the x-target. Again, the kicker features are the most important. The difference is that the y-positions are important for predicting the y-target of a dribble. These top twenty features account for 73% of the importance. Therefore, the other 27% belongs to the 581 other positive features.

Feature name	Relative Importance (%)
p_r_9_pos_y	-5.22
p_r_10_pos_y	-4.93
dribble_angle_8	-4.14
p_l_7_pos_r	-3.99
p_l_2_angle_goal	-3.70
p_r_11_pos_y	-3.45
p_r_7_pos_y	-3.24
p_r_2_kicker_r	-3.01
dribble_angle_3	-2.95
p_l_11_pos_y	-2.89
p_r_5_pos_r	-2.88
p_l_7_pos_y	-2.86
p_r_3_pos_r	-2.78
p_r_5_pos_y	-2.72
p_r_3_pos_y	-2.62
p_l_6_angle_goal	-2.33
p_l_9_pos_y	-2.33
p_r_3_kicker_r	-2.12
p_r_7_kicker_r	-2.05
p_l_5_angle_goal	-2.05

Table 4.22: Twenty features with relative most negative importance on y-target of dribble prediction

The most negative features for predicting the x-target is shown in table 4.22. The importances indicate that the "position" features are having a negative impact on the prediction of the y-target dribble. These twenty features account for 62% of the negative importance.

Chapter 5

Discussion

The pass classification prediction model performed worse than expected. The best pass target player classification model reached 82% accuracy, which is 2% lower than the performance reported in (Zare et al. 2022). However, the results may not exactly be reproducible since differences in the data generation method can produce differences in the training datasets. The best pass regression model is able to explain 95% of the variance in the x-target and 81% in the y-target (based on adjusted R^2-scores. However, the MAE of 4.91 for the y-target makes the predictions less useful to possibly exploit during matches.

The best-performing shot model can predict the agents' shot target accurately in most cases. This best shot model is trained without the features of teammates, and features with negative importance are filtered. The highest adjusted R^2-score of 93% implicates that the model can explain 93% of the variance in the agents' shot target with these features. In addition, an MAE of 0.58 is achieved. This MAE is inflated since the median absolute error is 0.093, and 95% of the shot predictions had a lower error than 2.5. The difference of 0.5 between the median absolute error and MAE results from the rare large maximum errors: when the model predicts the target shot near the right goal post (y-target $>$ 6), while the actual shot was near the left goal post (y-target $<$ -6), then the absolute error of this prediction is 12. However, in total, the adjusted R^2-score of 93% of the best performing shot model is 7% higher than the baseline, and the MAE decreased from 1.03 to 0.59.

The x-target and y-target of dribbles is predicted the most accurate with a R^2-score above 99% for both target variables. The best dribble model was trained with 707 features that had either positive importance for the x-target or y-target (or for both). The distance turned out to be harder to predict: the best model

reached an 82% R^2-score. However, the dribble distance was mainly used as an extra variable to measure the models' performance; the dribble distance is not included as a variable in the CooperativeAction object, which determines action that is executed by the agent. The high performance for the x-target and y-target predictions of dribbles might be explained by the relative simplicity of dribble behaviour in the agent2D team. However, the best dribble model improves the average adjusted R^2-score 5% (from 89% to 94%) compared to the baseline. Furthermore, the average MAE decreases from 0.92 to 0.70.

The filtering of features with negative feature importance turned out to be an effective way to optimise the models for predicting the shooting and dribbling behaviour. The filtering of the teammates' features and the features with negative importance reduced the MAE from 0.69 to 0.58 for the shot model, and the adjusted R^2-score increased from 91% to 93%. Furthermore, the average MAE of the best-performing dribble model is reduced from 0.79 to 0.70 by filtering out the features with negative or zero importance. The adjusted R^2-score also increased by 2% with. This method had no effect for the pass classification and pass regression model.

The analysis of the best models' predictions points out that the performance mainly increases with the number of samples per category. Consequently, the behaviours of the attackers can be predicted more accurate than the behaviours of the midfielders. In addition, the dribble and shooting behaviours can not be predicted for the defenders since these players do not execute (enough of) these behaviours. However, the same distribution of samples per player group is also present during the matches so a high prediction accuracy is more important for the attackers than for the defenders.

The usage of full-state mode and agent2D team (as own team and as opponent team) to generate the training data are the main limitations of the produced results. The combination of these elements makes it difficult to determine whether the models are beneficial to use in actual games. The full-state mode is turned off in actual competitions, which decreased the classification accuracy up to 12% for the pass target player prediction, as reported in (Zare et al. 2021). This could give a similar performance decrease for the regressions models used for predicting the passing, shooting and dribbling behaviour. However, I have not been able to test this, which was caused by a lack of time since it required generating and pre-processing 16 extra datasets.
Despite the performance decrease when using full-state mode, Cyrus still uses the predictions of the pass target player as input for its unmarking algorithm and neck-

update algorithms. The costs of an incorrect prediction are not high; it could, for example, be communicated to the top-3 predicted teammates that they will be the pass target player. In this case, all three agents could execute their unmarking behaviour (although only one agent will receive the ball). The same could hold for predicting the dribble target location and shoot target: a general idea of the target location could be sufficient.
Finally, using the agent2D team as own team is a limitation of the results but inevitable since the source code of the competitive teams is not publicly available. The limitation is that the Agent2D team has less complex behaviour patterns compared to the teams competing in the competitions. Therefore, the behaviours could be more difficult to predict when these teams are used. Furthermore, the Agent2D team can not compete with the actual teams, which decreases the performance up to 20%, as reported in Zare et al. (2022).

Chapter 6

Conclusion and future work

This thesis investigated how accurate deep neural networks can predict agents' passing, shooting and dribbling behaviour in a Soccer Simulation 2D match by shifting from classification models to regression models. The shooting and dribbling behaviours of agent2D soccer agents can be predicted accurately when the training data is generated with fullstate mode, and when the agent2D team is used as opponent. An adjusted R^2-score of 93% is reached with the best shot prediction model, and an average adjusted score of 94% is reached with the best dribble prediction model. These R^2-scores improved the baseline with 7% and 5% respectively. The pass regression model reached an average adjusted R^2-score of 88% but did not improve the baseline.
In addition, the importance of features for the models' shooting and dribbling predictions is analysed, and the effect on the models' performance by filtering out unimportant features is investigated.
Future work includes using the Data Extractor module for predicting the numerical features of the behaviours of soccer agents that have more complex behaviour patterns. This can be achieved by embedding the Data Extractor in a more competitive team. Furthermore, noise can be added to the observations to measure the impact on the regression models' performance. Next to this, a more competitive team as opponent team can be used to generate data when the own team can compete with it. If the results are promising, then the regression models' predictions could possibly be used as input for new algorithms to improve the players' and team game play. For example, the shot target predictions could be used for updating the neck angle towards the goal before receiving the ball, such that the kicker can already see the part of the goal where the actual shot is targeted to increase its accuracy. Furthermore, the shot target prediction could be used for a new algorithm to calculate the chance of scoring to improve the agents' decision-making.

Bibliography

Akiyama, H., Aramaki, S. & Nakashima, T. (2012), Online cooperative behavior planning using a tree search method in the robocup soccer simulation, *in* '2012 Fourth International Conference on Intelligent Networking and Collaborative Systems', IEEE, pp. 170–177.

Akiyama, H. & Nakashima, T. (2013), Helios base: An open source package for the robocup soccer 2d simulation, *in* 'Robot Soccer World Cup', Springer, pp. 528–535.

Budden, D. M., Wang, P., Obst, O. & Prokopenko, M. (2015), 'Robocup simulation leagues: Enabling replicable and robust investigation of complex robotic systems', *IEEE Robotics & Automation Magazine* **22**(3), 140–146.

Chen, M., Dorer, K., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Murray, J., Noda, I. et al. (2001), 'Robocup soccer server – manual for soccer server version 7'.
URL: *https://sourceforge.net/projects/sserver/files/rcssmanual/manual-7.08.1/manual.pdf*

Chicco, D., Warrens, M. J. & Jurman, G. (2021), 'The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation', *PeerJ Computer Science* **7**.

Kitano, H. & Asada, M. (1998), 'The robocup humanoid challenge as the millennium challenge for advanced robotics', *Advanced Robotics* **13**(8), 723–736.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E. & Matsubara, H. (1997), 'Robocup: A challenge problem for ai', *AI magazine* **18**(1), 73–73.

Leal, K., Pinto, A., Torres, R., Elferink-Gemser, M. & Cunha, S. (2021), 'Characterization and analyses of dribbling actions in soccer: a novel definition and effectiveness of dribbles in the 2018 fifa world cup russiatm', *Human Movement* **22**(1), 10–17.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436–444.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. (2017), 'Hyperband: A novel bandit-based approach to hyperparameter optimization', *The Journal of Machine Learning Research* **18**(1), 6765–6816.

Noda, I. (1995), Soccer server: a simulator for robocup, *in* 'JSAI AI-Symposium 95: Special Session on RoboCup', Citeseer.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L. et al. (2019), 'Kerastuner', `https://github.com/keras-team/keras-tuner`.

Willmott, C. J. & Matsuura, K. (2005), 'Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance', *Climate research* **30**(1), 79–82.

Zare, N., Amini, O., Sayareh, A., Sarvmaili, M., Firouzkouhi, A., Matwin, S. & Soares, A. (2021), Improving dribbling, passing, and marking actions in soccer simulation 2d games using machine learning, *in* 'Robot World Cup', Springer, pp. 340–351.

Zare, N., Sarvmaili, M., Sayareh, A., Amini, O., Matwin, S. & Soares, A. (2022), Engineering features to improve pass prediction in soccer simulation 2d games, *in* 'Robot World Cup', Springer, pp. 140–152.

Chapter 7

Appendix

7.1 Dataset information

Description:	Feature format, group	U	V	W
Position of players, 88 features	p_U_V_pos_W, position group	l,r	1-11	x, y, r, t
Player position relative to the kicker, 88 features	p_U_V_kicker_W, kicker group	l, r	1-11	x, y, r, t
Velocities of players, 88 features	p_U_V_vel_W	l, r	1-11	x, y, r, t
Unique abilities of players, 198 features	p_U_V_player_ type_W	l, r	1-11	decay, kickable, margin, dash_rate effort_max /min speed_max , size, kick power
12 ball features	ball_U_V,	kicker, vel, pos	x, y, r, t	/
Identifies distance to nearest opponent in 12 angles	dribble_angle_V,	0-11	/	/

Table 7.1: First part of the feature descriptions. "U", "V" and "W" describe the possible values the feature can take

Description:	Feature format, group	U	V	W
Identifies distance to nearest opponents in 12 angles	dribble_angle_V,	0-11	/	/
agent body, face view 44 features	p_U_V_W	l, r	1-11	body, face
player information, 110 features	p_U_V_W	l, r	1-11	card, side, unum, tackling, kicking
Team information, 22 features	p_l_U	is kicker, in_offside	/	
distance and angle to the goal, 33 features	p_l_U_V_goal_W	1-11	open, angle	angle, center_r, center_t
two riskiest opponents, 110 features	p_l_U_pass_opp_V_dist_W	1-11	1, 2	none, proj, diffbody, line, angle
counts, 44 features	p_U_V_count	1-11	pos, vel	
two nearest opponents, 66 features	p_l_U_near_V_opp_W, nearest opponents	1-11	1, 2	dist, diffbody, angle

Table 7.2: Second part of the feature descriptions. "U", "V" and "W" describe the possible values the feature can take

7.2 Pass prediction results

Table 7.3 displays the effect of filtering features based on their feature importance. The feature importance filter column describes which features are selected, and the "features" section describes the number of features selected with this filter. For example, the filter "type and player > 0" means only the features with positive importance for both the type and player prediction are selected, which results in 526 features. The table shows that the feature filters have the same performance as the best performing model shown in table 4.1.

Feature importance filter	Features	Player Accuracy	Type Accuracy
Type and player > 0	526	0.82	0.89
Type and player ≥ 0	683	0.82	0.89
Type or player > 0	745	0.82	0.89
Type or player ≥ 0	904	0.82	0.89

Table 7.3: Pass target model with uniform sorting and kicker first and features filtered

Feature importance filter	x-target MAE	y-target MAE	Average MAE
x-target and y-target > 0	3.07	5.09	4.08
x-target and y-target ≥ 0	3.18	5.27	4.22
x-target or y-target > 0	**2.95**	**4.88**	**3.92**
x-target or y-target ≥ 0	3.02	5.24	4.14

Table 7.4: MAE scores after filtering feature importances for x-target of pass and y-target of pass

Feature importance filter	Features	R2-score: x-target, y-target	Adjusted R2: x-target, y-target	Average R2-score	Average Adjusted R2-score
x-target and y-target > 0	526	0.95, 0.82	0.95, 0.82	0.88	0.88
x-target and y-target ≥ 0	683	0.95, 0.82	0.95, 0.82	0.88	0.88
x-target or y-target > 0	745	0.95, 0.81	0.95, 0.81	0.88	0.88
x-target or y-target ≥ 0	904	0.95, 0.81	0.95, 0.81	0.88	0.88

Table 7.5: Pass target player and type prediction performance with feature importance filters

feature name	Relative Importance (%)
p_l_2_pass_opp1_dist_proj	8.07
p_l_3_pass_opp1_dist_proj	3.92
offside_count	3.72
p_l_2_pass_opp1_dist_line	3.53
p_l_2_pass_opp1_dist	3.03
p_l_2_pass_opp1_angle	2.81
p_l_5_pass_opp1_dist_proj	2.64
p_l_4_pass_opp1_dist_proj	2.34
p_l_4_kicker_x	2.33
p_l_5_pass_opp1_dist_line	2.08
p_l_5_kicker_x	1.84
p_l_5_pass_opp1_dist	1.74
p_l_4_pass_opp1_dist_line	1.64
p_l_2_pass_opp2_angle	1.48
p_l_3_pass_opp2_dist	1.46
p_l_3_pass_opp1_dist_line	1.42
ball_vel_y	1.35
p_l_2_kicker_y	1.08
p_l_2_pass_opp1_dist_diffbody	1.05
p_l_3_pass_opp1_dist	1.02

Table 7.6: Twenty most important features for pass target player prediction

feature name	Relative Importance (%)
cycle	-5.49
ball_kicker_t	-4.75
p_r_9_kicker_r	-4.50
p_l_11_body	-4.42
ball_kicker_y	-4.01
p_r_6_pos_x	-3.52
p_r_3_pos_r	-3.03
p_l_3_angle_goal_center_t	-2.87
p_r_8_kicker_t	-2.78
p_r_5_player_type_decay	-2.70
p_r_2_pos_r	-2.38
dribble_angle_1	-2.38
p_r_8_pos_r	-2.38
p_l_3_vel_x	-2.29
p_r_7_body	-2.29
p_l_4_card	-2.29
p_r_9_kicker_x	-2.21
p_r_7_vel_r	-2.05
p_l_1_pass_opp1_dist_diffbody	-2.05
p_l_3_near2_opp_angle	-1.80

Table 7.7: Twenty most negative features for pass target player prediction

feature name	Relative Importance (%)
p_l_2_kicker_x	6.68
p_l_2_pass_opp1_dist_diffbody	4.99
p_l_2_angle_goal_center_r	4.86
p_l_11_pass_opp2_dist	4.52
p_l_2_pass_opp1_dist_proj	4.35
p_l_11_pass_opp2_dist_proj	3.87
p_l_10_pass_opp2_dist	3.39
offside_count	3.27
p_l_11_pass_opp1_dist	3.18
p_r_9_kicker_x	2.40
p_r_10_kicker_x	2.23
p_l_2_pass_opp2_dist_diffbody	2.22
p_l_2_pos_r	2.13
p_l_2_pos_x	1.74
p_l_9_pass_opp2_dist	1.56
p_r_11_kicker_x	1.21
p_l_2_pass_opp1_angle	1.10
p_l_10_pass_opp1_dist	1.07
p_l_9_pass_opp2_dist_diffbody	1.04
p_l_2_vel_r	0.99

Table 7.8: Twenty most important features for pass target type prediction

feature name	Relative Importance (%)
p_l_9_pass_opp2_angle	-8.81
p_l_9_pass_opp2_dist_line	-7.62
p_l_10_pass_opp2_angle	-7.16
p_r_6_kicker_t	-5.06
p_r_1_kicker_x	-4.41
p_l_10_pass_opp2_dist_line	-4.06
p_l_6_pass_opp1_dist_line	-3.30
p_l_1_vel_r	-3.22
p_l_1_in_offside	-3.19
p_l_6_pass_opp1_angle	-2.75
p_l_3_pos_y	-2.61
p_l_4_pass_opp2_dist	-2.25
p_l_1_body	-2.07
dribble_angle_5	-1.94
dribble_angle_11	-1.73
p_r_11_face	-1.66
p_l_3_kicker_t	-1.59
ball_vel_r	-1.54
p_r_3_vel_r	-1.54
p_r_1_vel_r	-1.52

Table 7.9: Twenty most negative features for pass target type prediction

feature name	Relative Importance (%)
p_l_3_kicker_x	5.64
ball_vel_y	3.53
p_l_2_kicker_y	3.18
p_l_4_pass_opp1_dist_proj	2.98
offside_count	2.78
p_r_11_kicker_y	2.62
p_l_3_pos_y	2.38
p_l_2_kicker_x	2.32
p_l_3_pass_opp1_dist_proj	2.01
p_r_10_kicker_y	1.93
p_l_2_pass_opp1_dist_proj	1.92
p_l_2_pos_y	1.84
p_l_10_kicker_y	1.83
p_l_1_face	1.82
p_l_2_pass_opp1_dist_diffbody	1.53
p_r_8_kicker_y	1.50
p_l_3_in_offside	1.45
p_r_9_kicker_y	1.37
p_l_9_kicker_y	1.30
p_l_4_pass_opp2_angle	1.23

Table 7.10: Twenty most positive features for pass y-target prediction

feature name	Relative Importance (%)
p_l_4_kicker_t	-4.05
p_r_4_kicker_x	-3.68
p_l_2_kicker_t	-3.35
p_l_9_pass_opp1_angle	-2.62
p_l_7_pass_opp2_angle	-2.60
p_l_9_pass_opp1_dist_line	-2.36
p_l_10_pass_opp2_dist_proj	-2.31
p_l_8_pass_opp2_dist	-2.13
p_r_6_pos_r	-2.10
p_r_1_kicker_x	-2.02
p_l_10_pass_opp2_dist	-1.98
p_l_8_pass_opp1_dist_diffbody	-1.91
p_l_8_kicker_t	-1.86
p_l_2_pos_t	-1.76
p_r_6_kicker_x	-1.75
p_l_11_body	-1.73
p_r_8_kicker_t	-1.61
p_r_7_kicker_r	-1.57
p_l_9_pass_opp1_dist	-1.56
p_l_6_pass_opp2_dist_diffbody	-1.55

Table 7.11: Twenty most negative features for pass y-target prediction

7.3 Shot prediction results

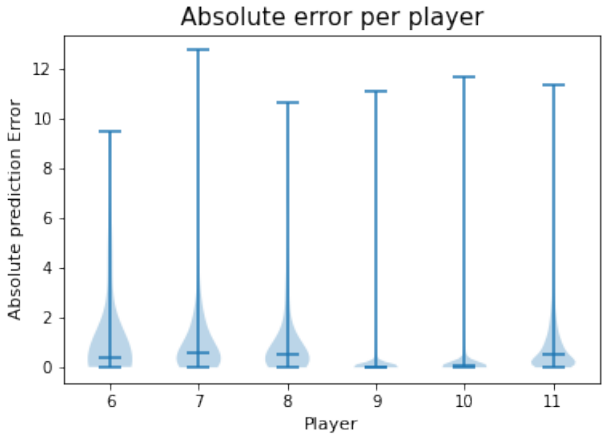


Figure 7.1: Violin plot per player, x-sorting

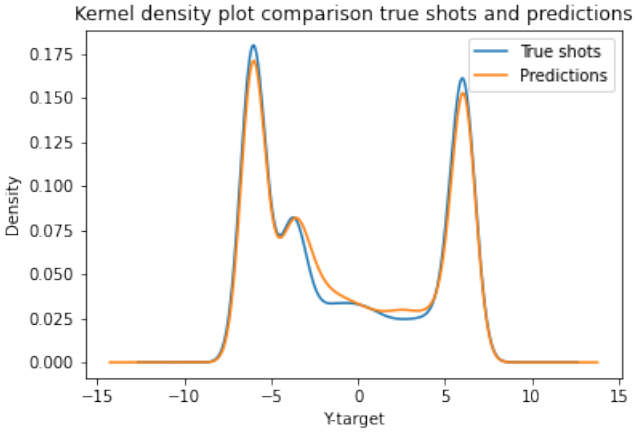


Figure 7.2: Density comparison, x-sorting

Y-target	Prediction	Absolute error	Kicker
6.108	6.071	0.037	10.0
-6.103	-6.045	0.058	9.0
-4.997	-3.974	1.023	9.0
-3.689	-3.406	0.283	11.0
-1.663	-1.202	0.461	11.0
5.998	5.995	0.003	10.0
-6.112	-6.122	0.01	9.0
-6.168	-6.099	0.069	9.0
-3.702	-3.347	0.355	11.0
2.651	3.678	1.027	11.0
3.766	4.112	0.347	10.0
6.045	6.095	0.051	11.0
-2.666	-3.466	0.8	11.0
4.146	2.514	1.632	11.0
5.994	6.003	0.009	10.0
-1.204	-1.187	0.017	11.0
-5.997	-6.025	0.028	9.0
6.041	6.04	0.001	10.0
6.095	6.097	0.002	10.0
-5.995	-6.062	0.067	9.0
-6.111	-6.029	0.082	11.0
-6.099	-6.076	0.023	9.0
3.681	1.605	2.076	11.0
-3.717	-3.651	0.066	11.0
-6.142	-6.085	0.057	9.0
2.824	3.383	0.559	9.0
6.161	6.159	0.002	10.0
-3.737	-3.452	0.285	11.0
-6.013	-6.005	0.008	9.0
6.013	6.077	0.064	10.0
-6.106	-6.115	0.009	7.0
-5.997	-5.999	0.002	9.0
6.106	6.108	0.002	10.0
4.592	3.765	0.827	9.0
-6.118	-6.077	0.041	9.0
-2.66	-2.192	0.468	11.0
5.996	6.017	0.021	10.0
6.141	6.19	0.049	10.0
-5.856	-0.315	5.541	7.0
-0.274	-2.135	1.861	10.0
2.7	0.816	1.885	11.0
6.125	6.152	0.026	10.0
5.995	6.144	0.149	10.0
6.008	6.006	0.002	10.0

Table 7.12: 45 shot predictions of the best model

7.4 Dribble prediction results

x target	x prediction	y target	y prediction	Distance	Distance prediction
27.692	27.918	-32.51	-32.227	3.22	3.347
18.54	18.957	-31.721	-31.636	2.449	2.829
50.758	51.081	14.672	14.739	2.186	2.157
1.425	1.794	-27.538	-26.001	6.003	4.662
48.218	48.551	9.821	10.108	2.443	2.813
21.467	21.245	31.7	31.626	2.868	2.915
-0.265	-0.091	29.8	29.692	4.721	5.019
15.885	17.443	-32.885	-32.1	1.773	2.996
-6.176	-6.795	22.748	23.341	5.739	5.728
-12.292	-11.566	26.798	26.562	6.57	6.293
48.472	50.637	-22.132	-21.073	1.795	2.42
48.964	49.986	-23.596	-24.224	3.31	3.318
-1.981	-1.915	30.899	30.658	4.982	5.252
32.918	33.482	-31.865	-32.031	4.493	4.379
49.506	50.106	-24.514	-25.282	2.368	2.417
3.111	3.982	30.372	30.149	2.453	4.232
18.975	18.599	-32.793	-31.914	3.531	2.849
22.375	19.029	-30.945	-29.597	10.927	7.072
50.159	50.281	21.429	22.046	3.088	2.486
35.691	35.963	-28.704	-29.023	3.514	3.662
3.791	3.011	1.771	2.148	4.755	4.33
-7.946	-8.484	-21.125	-20.744	5.357	5.521
45.279	47.233	14.828	15.808	7.125	7.626
51.476	51.332	-15.855	-16.139	1.792	1.82
8.722	8.755	31.114	31.091	0.0	0.356
6.56	6.799	-30.459	-27.609	3.693	3.621
49.687	49.923	-21.403	-21.256	0.0	0.114
-8.344	-9.166	-26.991	-27.615	6.524	5.695
10.467	11.937	-31.13	-30.509	0.214	0.377
46.302	47.908	30.209	29.685	3.588	5.192
11.856	12.387	23.46	22.211	4.685	3.75
13.283	12.118	31.33	31.107	3.325	3.42
49.948	49.066	-30.117	-30.346	10.473	8.678
18.509	18.664	-32.567	-31.725	3.408	3.15
49.653	50.453	22.091	22.635	2.524	2.714
3.669	3.517	-30.546	-30.208	0.083	0.041
50.53	50.607	-19.54	-21.183	3.807	2.441
21.738	21.989	-31.233	-31.904	3.337	2.905
4.616	4.411	1.392	4.519	3.015	2.822
49.582	50.884	24.978	25.126	10.275	10.63

Table 7.13: 40 dribble predictions of the best model