Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Csaba Juhász

SOCCER ACTION EVALUATION USING MACHINE LEARNING ALGORITHMS

SUPERVISOR

László Toka, PhD

BUDAPEST, 2021

HALLGATÓI NYILATKOZAT

Alulírott **Juhász Csaba**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2021. 12. 10.

...…….
Juhász Csaba

Összefoglaló

Az elmúlt években a labdarúgás egy több milliárd dolláros üzleti ággá nőtte ki magát: a játékosokat rendkívül magas pénzekért vásárolják a csapatok, a szponzorok bármit megadnak, hogy felkerülhessenek a csapatok mezeire vagy stadionjaira, míg a szurkolók is egyre komolyabb jelenléttel bírnak a klubok életében. A rajongók sikeréhsége kielégítése mellett figyelni kell a csapat anyagi korlátaira, így a klubok elkezdtek új megoldásokat keresni, hogy javítani tudják a teljesítményeiket mind a pályán, mind azon kívül.

Futball szurkolóként és mérnökinformatikus hallgatóként, mindig is nagy érdeklődéssel figyeltem a legutóbbi trendeket és megoldásokat a két terület kereszteződésén. A nézők támadó játékosok iránti elfogultsága már egy ideje kifejezetten foglalkoztatott. Mivel a meccseket gólra játszák, valahol érthető ez a hozzáállás, de én hiszek abban, hogy a védő és középpályás játékosok legalább olyan fontosak a csapat győzelmi esélyei szempontjából.

Így ennek a gondolatnak a mentén a szakdolgozatomban egy olyan gépi tanulási modelt hoztam létre a rendelkezésre álló adatok és technológiák használatával, ami nem csak a gólokat közvetlenül megelőző eseményeket tudja kiértékelni. Egy ilyen model később pedig a szurkolóknak és a szakértőknek is segíthet, hogy könnyebben megítélhessék egyes játékosok, vagy akár csapatok teljesítményét. Az eredményeim még tisztább megértése érdekében létrehoztam egy webes felületet is, amin vizualizálni is lehet őket.

Abstract

In recent years, soccer has become a business worth billions of dollars: players are being sold for previously unthinkable prices, sponsors are fighting tooth and nail for any chance to appear on the jerseys and stadiums of the leading teams, while the support from fans has never been as strong as it is now. To satisfy the ever-growing thirst for success from the supporters while also conforming to their monetary constraints, clubs have turned to new means of improving their performances both on and off the pitch. This resulted -among others- in the introduction of data science to professional soccer.

As a soccer enthusiast and a computer science student myself, I have always had a huge interest in the latest trends and solutions in this crossroad of these two fields. In particular, it has been the bias in the fans' perception towards rating attacking players that really caught my attention. As the game comes down to who scores the most goals in the end, this attitude is somewhat understandable, however I firmly believe, that the contributions of midfielders and defenders are just as important to a team's winning chances.

As such, I set out to elaborate on this train of thought, and using available technologies and data sets, create a machine learning model that would be capable of evaluating any situation on the soccer pitch, even those that were not directly followed by a goal. A model like this could in turn allow fans and experts to better judge certain players and even whole teams. To further help understand my findings, I created a web application, that can visualize my work.

Table of contents

1. Introduction

With the constant progress in processing capabalities of computers, the recent decades have brought significant forward leaps in several areas of computer science, among those are computer graphics and visualization, cryptography, or computer networks, to name a few obvious examples. In addition to these advancements, new fields have also been created that do or at least have the potential to help our everyday life tremendously. One of the most influential relatively recently conceived branches is Data Science. In the last couple of years this term and related expressions, like Big Data have gone on to carry a connotation of hollow buzzwords, but it is important to note, that in fact the area already has massive impact on our lives and its the companies that abuse the meaning of it. This process has culminated in the fact that nowadays data, and in particular personal data has become more valuable than ever and corporations such as Google, Facebook or Apple have managed to keep themselves successful by exploiting the great amount of information that they can extract from their users and the world around us. With the help of this knowledge, companies are able to create sophisticated algorithms to target audiences with personalized advertisements, oversee and handle financial developments with greater care and produce software and utilities for end users that make their lives easier.

Historically speaking, there has not been a great interest in the previous centuries when it comes to data analytics or statistics in sports. To discover the potential in some revolutionary techniques, competitors did not need to dig deep into the numbers: the superiority of the tumble turn in swimming, the effectiveness of the offside trap in soccer or the received edge when using the ever so slightly improving methods to start when running could quickly and easily become apparent to sports people. However, these observations could only help to discover the most notable innovations and as such after a while sports experts needed to resort to more and more refined metrics to find new ways to improve; with the aforementioned data revolution their work has become easier.

Motivation

UEFA Euro 2020 (that was held in 2021 due to the COVID-19 pandemic) had a live unique reach of 1.9 billion people during its one month run, making it one of the most watched sporting events in history. This should not come as a surprise, as soccer is one of the most popular sports globally, attracting significant numbers of followers from all continents. As a consequence, it has grown to be an industry worth billions of dollars, if not more. The current record for the highest fee paid for a single player was set in 2017 when Paris Saint-Germain acquired the Brazilian Neymar for around 250 million dollars. Owners of clubs and national association expect investments of this excess to pay off in some form, either by revenues generated through shirt sales, TV deals and advertisements or through success on the soccer pitch that usually translates to higher monetary rewards from the leagues. To achieve these latter goals, teams need to make informed decisions when it comes to signing new players or releasing those that do not contribute enough to the team, they have to prepare for their opponents' tactics adequately, and prevent potential injuries to their stars while keeping up their fitness levels.

When focusing on on-pitch performances, it is crucial that teams have objective metrics that measure contributions, when evaluating players. Clubs from lower divisions or those, that can not afford the personnel to analyze, usually either only rely on what they see during matches, or in better cases only consider simple attributes, like goals scored, tackles attempted, or headers won. Using these sorts of data points is not necessarily considered bad practice, and definitely does help somewhat, but it also needs to be acknowledged, that these features do not tell the whole story about a player's actions. When analyzing the number of goals scored, we need to ask ourselves: is the amount achieved sustainable? This means: do they usually score from situations, where they have high chance of finishing, or are most of their goals flukes, that come from 20-30 meters away from the goal? Is their number of goals that high or low thanks to their great/poor chance creation or is it rather due to their above/below average finishing? Are they consistent in front of goal, or do they have droughts during seasons, when they go weeks or months without getting on the score sheet? Similar questions can be asked for these sorts of primitive data points. These issues can be tackled using more refined data sets, that include attributes such as expected goals, that tells us the

aggregated quality of chances that a player (or team) has had during a match or season, centrality, that intends to show players' contribution to their teams' passing play, or a pass map, that helps us understand the potential deficiencies in the team play. An example for this sort of analysis can be seen on figure 1. A pass network shows the average positioning of all players during a game and the frequency of passes between players, the thicker the line between two players, the more they passed to each other. Even from relatively simple figures like this, we can easily draw significant conclusions such as the fact that Roberto Firmino, who is Liverpool's striker on paper, operated more as an advanced playmaker in midfield and did not manage to really transition the ball to the forwards (Mané and Salah). We can also see that the team in general suffered to progress the ball towards the goal, as most of the passes occurred between the defenders and even if the ball got high up the pitch, it was usually closer to the flanks than to the center of the pitch.

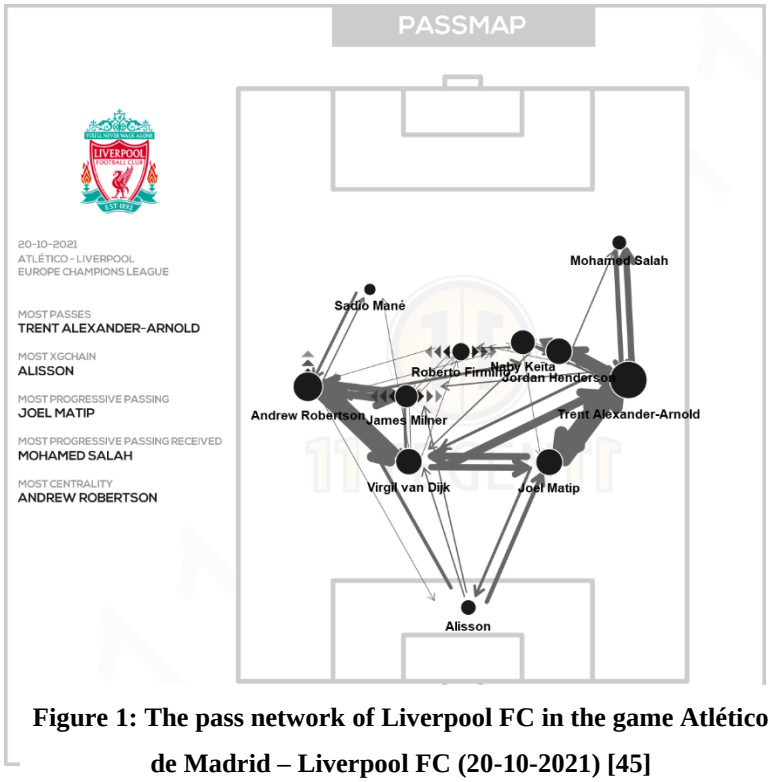


Figure 1: The pass network of Liverpool FC in the game Atlético de Madrid – Liverpool FC (20-10-2021) [45]

As an enthusiastic soccer fan, these questions interest me on a personal level. Even though, as my research into related work below shows, several experts have tried to tackle this area, I believe that these solutions sometimes are not comprehensible for everyday followers of the sport, who at the end of the day are still the ones who this industry is supposed to be aimed for. As such, my intention with this project is to on one

hand create a data science model that could potentially live up to or in some way extend the models put forward by professionals and on the other hand make my findings accessible for people that are not familiar with the mathematical and computer scientific background required for these algorithms. I believe that a web site which visualizes these data can help their understanding tremendously. In order to meet these expectations, it is crucial that I have a solid grasp of the underlying mathematical principles of the existing machine learning techniques, while also being able to engineer and prepare the existing data in the correct manner. Once I have succeeded on that side of the project, I will need to have the skillset to design and implement a web application, that can create and present an understandable visualization given some parameters.

2. Relevant literature and related work

During my work, I sought to progress in a way that is both measured and organized. In the area of data science, the leading approach for problem solving is CRISP-DM, which stands for Cross Industry Standard Process for Data Mining [1]. This methodology provides a stable basis for researchers and personalities in the industry to carry out such a project in a standardized way. The represented Data Science life cycle is made up of six stages:

- Business understanding

  We need to have a firm knowledge of what exactly the clients expect from our work. Having achieved that, an exact goal has to be defined for the project, which requires an accurate assessment of the technologies and resources that are available for us. In my case, this involved deep discussions with my consultant about both of our expectations and then coming to an agreement accordingly.

- Data understanding

  This step focuses on the data that will be used. If not already available, data needs to be acquired in some form, either through the clients, or through third-party resources. We then analyze and explore the data, looking for potential inconsistencies or missing values, understanding its structure and potential relationships within it. This phase of my work will be discussed in detail in the 3rd paragraph.

- Data preparation

  Now that we are fully aware of our data's main attributes, deficiencies, and strong suits, we have to make the most of it, so that our models get as much and as accurate information as possible. We have several options, to accomplish that:
  - Deciding which parts of our dataset to include and exclude
  - Solving those inconsistencies and errors in our data that we have discovered earlier
  - Deriving attributes and values from our data that might be more relevant for our problem at hand

- Integrating different datasets to create more massive and robust ones instead
- Formatting our data in a way that it becomes comprehensible for models

According to experts, this stage makes up at least 50% of the actual work [2]. This phase of my work will be discussed in detail in the 4th paragraph.

- Modeling

  Having done the previous steps, we can finally get down to selecting an adequate algorithm to process our data, splitting that data into training, test and validation sets and then building or training our model, with different hyperparameters and selecting the best performing one. This phase of my work will be discussed in detail in the 5th paragraph.

- Evaluation

  This stage serves the purpose of reviewing our project. We determine whether we have met the requirements set up in the Business Understanding stage, reflect on possible mistakes we might have made along the way and start thinking about potential improvements on our solution. This phase of my work will be discussed in detail in the 6th paragraph.

- Deployment

  The finished product still needs to be presented in some form to the customers. In some cases, it happens in the form of back-end integration, but as I will discuss in the 7th paragraph, I have chosen to visualize my findings on a web site.

As seen above, I aim to structure my thesis in an analogous way to the CRISP-DM methodology.

2.1 Sports analytics and popular culture

When discussing the history of sports analytics and the data driven approach towards player recruitment and tactical analysis, some consider the Oakland Athletics baseball team to be the first revolutionaries of the area [3]. Until 2002, their new attitude towards scouting had been unthinkable: instead of solely relying on human judgement on how a player performs on the pitch, they set out to use statistical input as well to get a full picture regarding certain targets. The experiment was a huge success:

even with their significant budgetary constraints, they managed to compete with the strongest teams in the league. The publication of the book *Moneyball: The Art of Winning an Unfair Game* [4] in 2003 that goes into great detail as to how they achieved this level of accomplishment has become a "must-read" for those, who are interested in the area, and it really helped spread the notion that facts and numbers can tremendously help professional clubs and experts.

Talking strictly about literature related to soccer analytics, there probably has not been any piece of work published that has had the same effect on the popularity of this approach. Nonetheless, *The Expected Goals Philosophy: A Game-Changing Way of Analyzing Football* [5] and *Football Hackers: The Science and Art of a Data Revolution* [6] are definitely well-known books in this circle, that do tackle this issue in detail and have certainly served as an example for my work too.

2.2 State of the art

While popular culture certainly has had an effect on the demand for sports and soccer analytics, it is really important to take a deeper dive and examine where the science lies right now in particular when it comes to action evaluation. Doing that allows me to establish a stable starting point, make more informed decisions and as such create a model that satisfies my expectations.

> *"Despite many recent innovations, most advanced metrics […] remain based on simple tallies relating to the terminal states of possessions […]. While these have shed light on the game, they are akin to analyzing a chess match based only on the move that resulted in checkmate, leaving unexplored the possibility that the key move occurred several turns before." – D. Cervone et al.* [7]

Although the quote above has been written in relation to basketball, I believe that the same holds true for soccer as well. There is a tendency to overrate the importance of shots and goals at the expense of dribbles, passes or other movements that lead to those dangerous actions in the first place. As a consequence, I set out to seek solutions that keep this concept in mind and approach the problem accordingly.

A great source of knowledge regarding machine learning based soccer analytics has been the Katholieke Universiteit Leuven's (KU Leuven) research. The university is considered among the best in this area and fortunately have also discussed action evaluation in some form in several papers [8]. As an example, I have learnt a great

amount from *Actions Speak Louder than Goals: Valuing Player Actions in Soccer (T. Decroos, et. al.)* [9] and *VAEP: An Objective Approach to Valuing On-the-Ball Actions in Soccer (T. Decroos, et. al.)* [10]. These pieces of work introduce the topic of action valuation and its considerations in an understandable yet comprehensive way. They establish the fact that certain situations must be looked at in context, as for example a back pass from a defender to their goalkeeper means definitely a lot less in terms of goal-scoring probability than a through ball from a playmaker to a winger up the pitch even though they are both classified as passes in most datasets. Lots of models fail to take into consideration these differences and as such perform worse when predicting the probability of a goal. The writers of the paper have chosen 151 different attributes to base their predictions on, these range from location data to body parts used or the current standings of the game. Core element of creating a machine learning model is an adequate decision when it comes to choosing the learning algorithm and as we can see from their results, CatBoost, Logistic Regression, XGBoost and Random Forest all seem to be realistic options, that I will have to further examine. In their case CatBoost was determined to be capable of achieving the best results. A significant difference between the solutions presented in the papers and my project is the fact that my work is solely interested in the goal-scoring chances of the attacking team, while theirs also focuses on potential negative effects of an action and considers whether the opponent managed to score in the following seconds.

Even though the work of K. Singh is not applicable directly in my case, I believe that his article, *Introducing Expected Threat (xT)* [11] needs to be mentioned. His piece has been cited in several of KU Leuven's papers (including the aforementioned ones), and for a good reason. Singh uses a unique method in that he divides the pitch into 192 zones and tries to estimate a danger value for each one of them. This approach is based on the notion that wherever on the pitch the player is with the ball they are faced with two options: either try and shoot at the goal or try and advance the ball. In simple terms, knowing the probability of converting the shot from that position to a goal and the matrix of probabilities representing the chances of advancing (via pass or dribble) from a given zone to another one (which are both easy to calculate given historical data) will allow us to calculate how likely it is in each zone that the team will end up scoring.

Decomposing the Immeasurable Sport: A deep learning expected possession value framework for soccer (J. Fernández, et al.) [12] introduces a completely opposing

way of looking at action location on the pitch. It is important to notice that the positioning of the opposing team should greatly influence our judgement of where the ball is possessed. To name an example, against teams, that “park the bus” (play with very low lines of defense), it is relatively easy to get close to the goal, however that does not mean much when there are still 6-7 players left to get past, while high-pressing clubs will usually make it harder to advance towards the goal, when the attacking team does get close though, it will usually be easier to take a shot on goal. The writers define relative location, which describes the attacking players’ position in relation to the opponents’ defensive lines. I find this approach highly interesting and believe that this attribute can further improve my model.

As we can see, several different options have been proposed to counter the two greatest challenges of soccer analytics: the lack of available detailed data and the fact that in a game of more than 1000 actions only about 0-6 are usually goals. During my research I have found fascinating techniques even besides the previously mentioned ones. Most notably, Maaike Van Roy et al. suggest the creation of a Markov Model to predict the value of a given action [13], while Yudong Luo et al. [14] have implemented a deep learning-based solution.

3. Data understanding

With the recent data revolution in sports and specifically soccer, more and more companies have turned their attentions towards supplying clubs, journalists and sometimes even fans with details about matches, teams, and players. Naturally, the less one is willing to pay for this sort of information, the less will be made available for them. As a consequence, fans for example can only rely on some blog posts, low quality web sites or tweets to expand their knowledge in this area, while top clubs can afford to even buy companies that will gather and analyze data for them. A notable example for this is Arsenal Football Club, who have acquired the US based StatDNA [15] in the 2011-12 season. Several businesses operate independently that can then be approached by many clubs for consultation and data attainment purposes, the most significant ones include Opta – Stats Perform [16], Instat Sport [17], and Wyscout.

In this paragraph, I will summarize the way I have gotten hold of the dataset for my work, explain its background and structure and then detail how I made it compatible with my aims regarding my project.

3.1 Data acquisition

Mainly collaborating with professional soccer clubs, these aforementioned platforms offer their services for prices, that are not realistically affordable, for a university student writing their thesis. I have tried contacting Wyscout and Opta but with no luck, as Opta's response states: "[they are] not offering free educational/research data at this time, or for the foreseeable future". As a rather desperate measure, I have looked into scraping data from larger soccer related websites, such as Transfermarkt [18], which as its name suggests mainly deals with transfer values but also has relatively detailed statistics for players, WhoScored.com [19], a site that visualizes data purchased from Opta, or Football Manager, that is one of the most realistic soccer game out there, when it comes to scouting players [20]. Many great libraries have been created in Python to do this kind of job: Selenium is a great tool for web browser automation which lets us browse search results very easily, or Beautiful Soup that helps extract meaningful information from HTML files. I even ended up writing a Python utility to scan Transfermarkt using these libraries however I soon had

to realize that the data acquired this way is not enough to create an adequate model. However, a future extension of my work could include comparing players' prices to their performances according to my model and as such I kept these scripts for the moment.

Thankfully, through my consultant I got the chance to get in touch with a Hungarian sports analytics firm, Xfb. Analytics. They provided me with an InStat dataset containing data from 105 matches, from competitions such as the Champions League or the German Bundesliga.

3.2 Instat data

For each game in the dataset, I had two XML files, one for the play-by-play data and one for the fitness data (which means 210 files in total) with the following folder structure:

Root folder > [Team Name] > [Date Of The Game_Home Team_Away Team_Result] > fitness_[nr]_[Match ID].xml / markers_[nr]_[Match ID].xml

3.2.1 Play-by-play data

Sports analytics platforms like InStat usually gather and process so called play-by-play data, which means that every move (or sequence of moves) gets labeled and analyzed extensively. An alternative name for play-by-play data is marker data. These moves include obvious actions like passes or shots, but motion with the ball (dribble), set pieces or even blocked shots also get handled. For now, unfortunately this procedure can not be automatized, as current solutions are only able to locate and track players on the pitch, but the technology is not there yet to accurately analyze the type of action and its attributes that had just taken place. This means that this great amount of play-by-data available at our disposal was manually gathered by humans. Consequently, some level of error is to be expected, but not to an extent that it could have a serious impact on our model's performance.

Around 45 attributes belong to each action: it is not necessary to list all of them, but the following are the most important ones:

id: Unique identifier of an action
action_id: The type of the event (e.g. Goal, Ball Recovery, Pass To Offside Position)
position_id: What position the player in possession plays (e.g. Cent Back, Right

Winger)

opponent_position_id: What position the opposing player (if there is one) in possession plays

half: Which half of the game we are in (first or second 45 minutes)

second: How many seconds have elapsed since kick off

pos_x and pos_y: The coordinates of the event

pos_dest_x and pos_dest_y: In case of a sequence of moves the coordinates of the ending move

standart_id: The type of action (e.g. Free Play, Set Piece)

attack_status_id: The stage at which the attack currently is (e.g. Start, End, Attack)

body_id: The body part used for the given action (e.g. Right Foot, Head etc.)

3.2.2 Fitness data

The dataset I got hold of also contained location and fitness data for each match. As opposed to the play-by-play data, this sort of detail is provided for each player at each second of the game and while the former is created manually, fitness data is generated using videos captured by 2 full HD cameras on 2 sides of the pitch which then get analyzed by automatic algorithms and get turned into a 2D model of the match.

This dataset proved to be invaluable for my project for its following attributes:

pos_x and pos_y: The coordinates of the player at a given second

acc: The rate of acceleration (in m/s^2) of the player at a given second

speed: The speed of the player at a given second

As an interesting sidenote, I would like to mention that I also had some more fitness related attributes at my disposal, which include Energy Cost and Metabolic Power, but I ended up discarding these. The reason behind my decision was that these values seemed rather arbitrarily defined and did not consider the variance of stamina and physical shape of the players to any extent. The displayed amounts were not measured by any specific on-body devices but rather by an algorithm that calculated them the same way for each player. It is not hard to imagine however that a wing-back, whose job it is to keep running up and down the pitch will definitely have higher endurance than a non-ball-playing center back whose team is dominating most games.

3.2.3 Teams and competitions

The following teams' games from the 2017-18 season were made available for my work:

S.L. Benfica – 5 Champions League games
PFC CSKA Moscow – 4 Champions League games
Ferencvárosi TC – 20 Nemzeti Bajnokság I games
Feyenoord Rotterdam – 5 Champions League games
1. FSV Mainz 05 – 23 German Bundesliga games
FC Porto – 5 Champions League games
Qarabağ FK – 5 Champions League games
FC Schalke 04 – 24 German Bundesliga games, 1 friendly
FC Shakhtar Donetsk – 8 Champions League games
FC Spartak Moscow – 4 Champions Legaue games
Videoton FC (currently known as Fehérvár FC) – 17 Nemzeti Bajnokság I games

It is easy to conclude that the 3 competitions included are relatively diverse in terms of the criticality of results: Bundesliga and NB1 are long-term tournaments where bad performances can not get punished as much, whereas Champions League requires each game to be played with maximum focus as the group stages are made up of 6 games only and then the knockout stage involves two matches each round. The diversity is also apparent when it comes to the quality of the players represented: the team with the lowest market value in the Champions League was NK Maribor with 9.06 million Euros, which would make it the fourth strongest team in Nemzeti Bajnokság I, while Hannover 96, the cheapest team of the German Bundesliga was still worth 3 times as much as the most expensive team in Nemzeti Bajnokság I [21] [22] [23]. This sort of variance in competitions could help make my model more robust during training.

4. Data preparation

4.1 Tools

To process the dataset and then implement learning algorithms it is necessary to choose an adequate programming language with satisfactory utilities and libraries. For data science purposes, most experts recommend the use of Python, R, or Java [24]. Having had more experience in the former, I have chosen to work with Python on my project. Python is a general-purpose interpreted, interactive, and high-level programming language which was first released in 1991. The language was designed to be easily readable and to make the programming process easier even at the expense of runtime. The language supports functional, object oriented, procedural, and imperative programming. One of its many advantages is the broad collection of open-source libraries that help cover basically all areas of Computer Science.

A great tool for data manipulation in Python is Pandas, a library that, as its mission statement reads, “aims to be the fundamental high-level building block for doing practical, real world data analysis in Python” [25]. When it comes to my work, it was their DataFrame class, that helped me tremendously when manipulating my dataset.

4.2 Data cleaning

As I have discussed prior, location data is processed using computer vision methods, which does have the tendency of sometimes inaccurately tracking players and if two or more of them overlap at a given moment, it might not even register them properly. Thankfully, I did not find a significant number of anomalies, but I still aimed to make my learning data set as accurate as possible. Most incorrect values I encountered were Nans, which could be filled in either by interpolation or extrapolation. I decided that in this particular problem interpolation seemed like a more reasonable solution, due to the high number of changes in direction and speed during a game and as such with the help of built-in pandas methods, implemented linear interpolation.

The other inconsistency that needed fixing was related to the difference in the representation of player and event locations. Event positions were always calculated from the point of view of the attacking team (as in, who is in possession of the ball),

while the player position data used a coordinate system that has absolute coordinates with the origin lying in the intersection of the half line and one of the sidelines (-52.5 to 52.5 at x axis and 0 to 68 at y axis). To remedy this issue, I created a script that grouped each match by their two halves, checked where the teams' goalkeepers stood at the time of kickoff and from that could compute whether the coordinates needed inverting or not.

4.3 Data storage and integration

Next, I needed to incorporate these XML files into two coherent units (one for the marker data and another one for the location data) and then make them more manageable for machine learning purposes as their initial format is not really digestible for any sort of classification algorithm. I ended up creating a Python script that collects and processes all files and then create two DataFrames, where each XML attribute is represented as a column. Obviously iterating and reading through a list of 105 files every time I aim to teach my model is highly inefficient, and thus, I wrote these objects into two kinds of files. Reading a file with hundreds of thousands of rows is highly resource intensive which means storing it in an adequate format could save me a great amount of time. For this reason, I have chosen to store these DataFrames in parquet format. A more naturally sounding option would have been CSV, but having compared their speed and storage requirements, I have decided for parquet. Figure 4 illustrates the difference between the two formats.

Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet format*	**130 GB**	**6.78 seconds**	**2.51 GB**	**$0.01**
Savings / Speedup	**87% less with Parquet**	**34x faster**	**99% less data scanned**	**99.7% savings**

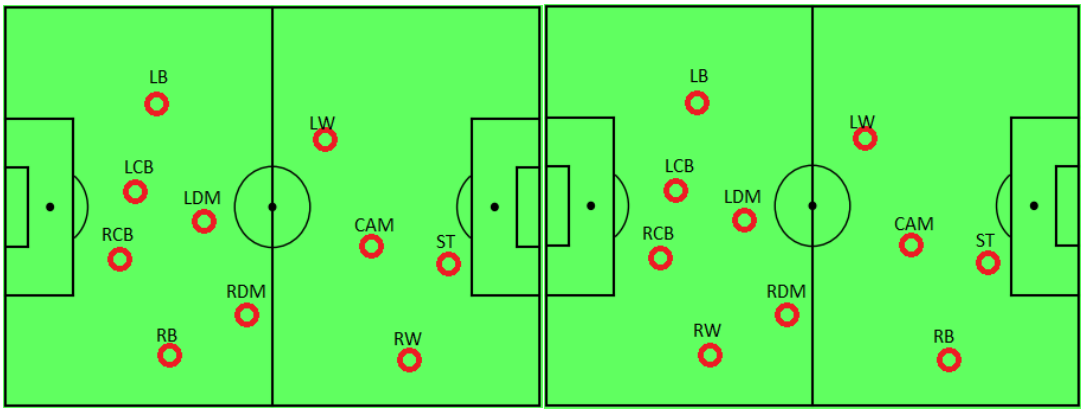
Figure 4: The performances of CSV and parquet files on 1 TB of data [46]

As Figure 4 suggests, there is a tenfold difference between storage space needed, while parquet also requires around 30 times less time to be processed as opposed to CSV. Having run my script, the marker data parquet ended up containing approximately 400 000 rows (records), while the fitness data parquet had around 14 000 000 rows.

While certainly not impossible, it is also relatively hard to input two separate DataFrames into machine learning models in Python, so my next task was to find a way to join these two sets. An obvious way to achieve this is by finding the corresponding fitness data to the second that an event occurred in and appending that information for

each player to the event data. This approach however can face some mild difficulties. First of all, the timer in the fitness data was counted from kickoff, while the timer in the event data was counted from the start of the video footage. This problem could easily be solved by checking the time of the first pass of the half and subtracting that amount from each timestamp. Some other small adjustments had to be made regarding the timings also due to the fact that the 'second' attribute in the marker data had a precision to a hundredth of a second, which meant, that they had to be rounded up or down to the nearest whole number in order to be synchronizable with the fitness data. These changes only required a couple of lines of code.

The datasets were now ready to be merged. One last issue to be figured out involved the difference between what I called the dilemma of the absolute and relative positioning of the players.



Figures 5 and 6: The dilemma of the absolute and relative positioning of the players

Figures 5 and 6 illustrate two almost completely identical situations on the pitch, the only difference being the fact, that the right back has taken the right wingers place in the attack, while the right winger tucks in, taking their place, so that the team does not get exposed on the counterattack. This situation (or one where they switch places with the defensive midfielders) is likely to come up nowadays with the more and more free roaming roles of the full backs, see Trent Alexander-Arnold's or João Cancelo's positioning for their respective teams. These two situations should be seen as basically the same, however if we were to just lazily create an attribute for the positioning of each player, a traditional machine learning algorithm would treat them very differently, since values or dimensions in vectors can not be simply exchanged. To tackle this problem, I sorted players by their distance to the opposition goal, so instead of creating attributes for concrete playing positions (as in goalkeeper, center back etc.), the new attributes

could always be compared by themselves as they always represented the locations on the pitch the best. As mentioned earlier, I did not consider health and shape data for the players, I only took location, speed, and acceleration. These new details in the marker data could help the model make a distinction between an action where an attacking player is ready to burst forward and go one-on-one against the keeper or one where no attacking players are playing that high up the pitch.

4.4 Preparing for neural networks

While traditional classification algorithms like decision trees, Bayesian classifiers, or support vector machines mainly expect vectors as inputs, neural networks, and in particular convolutional neural networks deal mostly with matrices or even higher dimensional data. The DataFrame that I generated however only consist of a great number of vectors. Obviously one-dimensional convolutional layers or linear layers can process vectors easily, however the essence of the algorithm lays in dealing with higher dimensional data, that is why deep neural networks are used for image recognition tasks. Inspired by the accuracy with which neural networks are able to classify certain images, I set out "imagify" my data, which meant creating 4 two-dimensional lists for each event of the games, they were separated by attacking and defending team and speed and acceleration. The lists, with size 105x68 represented the pitch and initially all values were filled with zeroes. Then I iterated through all players' attributes at a given second of the game grouped by attacking and defending teams, where speed_list[player_location.x][player_location.y] got the current player's speed, and acc_list[player_location.x][player_location.y] got the current player's acceleration. This basically created 4-channeled tensors with a size of 105x68. These parameters are comparable to that of RGB pictures, that have 3 channels (for red, green and blue values each) and have a size of usually at least 16x16, but 128x128 is a lot more normal.

The question of storage arose again, as CSV, parquet and even pickle files seemed to be at least 10 kBs in size which in the case of 400 000 events added up to 4 x 4 = 16 GBs of space required. Reading this amount of data each time I trained the model would have created a serious bottleneck, so instead of storing the whole list, I ended up creating a list with only the non-zero values given. The following structure was used:

[(player_1_x, player_1_y, player_1_speed), …, (player_11_x, player_11_y, player_11_speed)]

This way, I saved around 15.4 GBs of space and also expected lower reading times.

4.5 Deriving new attributes

While knowing the x and y coordinates of players is definitely an important aspect, we can not yet distinguish a player deep into the opponent's territory, that is just walking backwards slowly and as such not actively helping the attack or a teammate that is running at top speed towards the opposing goal, waiting to receive the ball. To fix this shortcoming of the data set, I created four new attributes: which direction the player is moving (x and y coordinates) their speed towards the goal and their acceleration towards the goal. Calculating the direction is straightforward: take the previous location of the player (one second ago) and subtract it from the current position. Figure 7 shows how the speed and acceleration values were computed. We need to calculate the vector from the player towards the goal and then calculate the length of the previously calculated movement vector's projection on that. This can be achieved by calculating the angle between the two vectors and calculating its cos value. Finally, this result needs to be multiplied by the speed and acceleration values respectively.

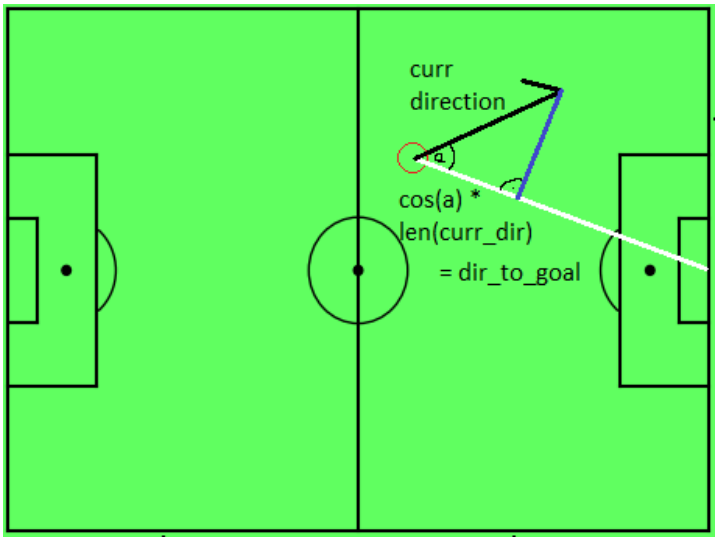


Figure 7: Calculating the length of directional vector towards goal

Defining and labeling dangerous sequences was the most critical one in the sense that the danger attribute of the data was the one the model would aim to predict.

Thus, I spent a significant amount of time to determine what the model should consider dangerous. In the end I decided that an event should satisfy one of two criteria: if the event is a shot on goal, it is automatically considered to be dangerous as it poses a direct threat of goal, or if an event's type is not a shot but happens in a 15-meter radius of the center of the goal, it is also labeled as dangerous. In the end I decided to add an extra condition: whatever the event, if the angle between its directional vector towards goal and the goal-line is less than 20°, we discard it, because shots from there can easily be saved by the goalkeeper. I concede that this definition of dangerous event is rather arbitrary, which is why I wrote the labeling method in a way that it can be parametrized freely, so that both the distance and the angle can be changed. After labeling the events as dangerous or not dangerous, I grouped all sequences of possession and checked each one whether it contained a dangerous event. If it did, I labeled all actions of the group as leading to dangerous event. I considered a sequence of events as part of one possession as long as no opposing player touched the ball more than once. Allowing one touch was necessary, because in my understanding of the game, just because a defender manages to slightly feather the ball, that will not start a new sequence of possessions, however InStat does note that touch as a unique event. When evaluating the output, I found approximately 100 dangerous possessions, where a sequence consisted only of a touch of the ball and a foul from an attacking player. I decided to discard those sequences and reran my script in a way that did not allow this sort of two-move outliers.

I had a hypothesis that maybe players' fatigue might also play a role in deciding the dangerousness of an event and while I still believe, that the InStat data is not that reliable, the time elapsed is a universal value in the sense, that on average each player will tire the same amount after the same amount of time.

4.6 Other utilities

As part of the preprocessing stage, I created some utility methods, that helped me manage the data better. For one, I needed a method that stripped the DataFrame off of unnecessary rows such as ones where the starting line-ups are shown, or summaries at the end of matches. Fortunately, this way of editing a DataFrame is very straightforward in pandas. Another helpful method chose feature names that would be used in the model in a very parametrizable way.

5. Modeling

During my work I have ended up using two different types of classification algorithms. In this paragraph, I will first describe the aim of a classification task and then building from the bottom up, I will demonstrate the mechanism of my chosen algorithms. Following that, I will introduce my concrete implementation of these methods.

5.1 Classification

The goal of a classification task in data science is to assign new data into predefined classes in a way that those classes remain as distinguishable as possible. In other terms, classification tries to find a function that puts discrete labels on the input variables [26]. There are several approaches to achieve this, all of them excel in different environments. During my work, tree-based solutions and neural networks showed the most promise among all algorithms, however in the initial phases I had plenty of options at my disposal. These include Bayesian classifiers, K-Nearest Neighbor classification, Logistic regression, and Support Vector Machines. For my particular task however, neither of them returned adequate results and as such I decided against using them any further.

5.1.1 Decision tree classification

Decision trees target to learn/create decision rules based on the training data. The creation of these rules is the result of a recursive algorithm in which:

1. Consider the set of attributes
2. Find out which attribute holds the most information about the classes (for example if a variable is greater than X, the assigned class will be A 90% of the time)
3. Create a rule based on that observation and split the data into two based on that rule
4. Repeat from 1. on the two newly generated subtrees until a) A leaf only contains members of one class b) We have reached a previously defined

maximal depth for the tree in which case we compare the number of samples from each class and assign to the leaf that class with the most representatives

As Figure 2 illustrates, after the decision tree has been trained, these rules can be used as if-else statements, so that when a new input is being processed, we always choose the branch that returns true for the input for that rule. When we reach a leaf node, we assign the corresponding class to our input. The greatest concern regarding decision trees is their tendency to overfit on training data.

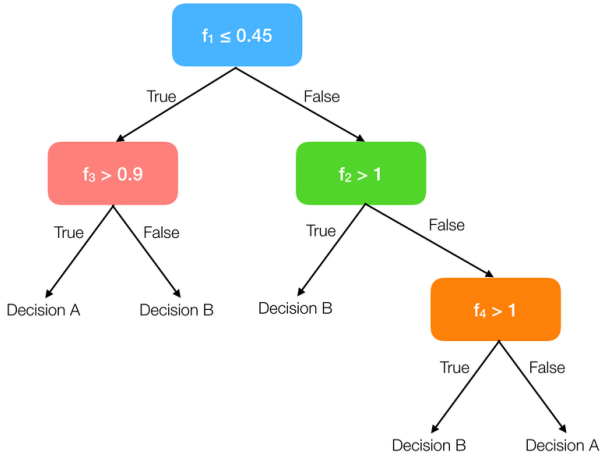


Figure 2: The structure of a Decision tree [27]

5.1.2 Random forest classification

Random forest classification is an ensemble learning method, which bases its results on the notion of the „Wisdom of the crowd" [28]. In this case this means that it creates multiple decision trees by slightly randomizing the training data for each tree. This randomization includes:

- Assigning 'bootstrapped' data to each trees' training samples, which is a random subset of the samples, where duplicates are allowed

- When building a given tree, at each node, only a random subset of attributes is made available for consideration

As a consequence, any single tree generated will likely perform worse when predicting classes, but their aggregated outputs will usually tend towards the correct label, and as a unit their most voted result will be accurate. Figure 3 demonstrates this algorithm. The reason why they have the potential to perform better than decision trees is that their individual errors and tendencies to overfit is completely counteracted by all the other trees in the forest. What needs to be kept in mind is that the more the trees are

uncorrelated, the more likely it is that overfitting will not be an issue, and that is why randomly choosing the attributes at each node of the decision trees is important. The fact that multiple trees need to be generated also means that these models usually take more time to build. Another difficulty when it comes to the usage of random forests is the fact that a trained model can not be interpreted as easily as a single decision tree, if at all.

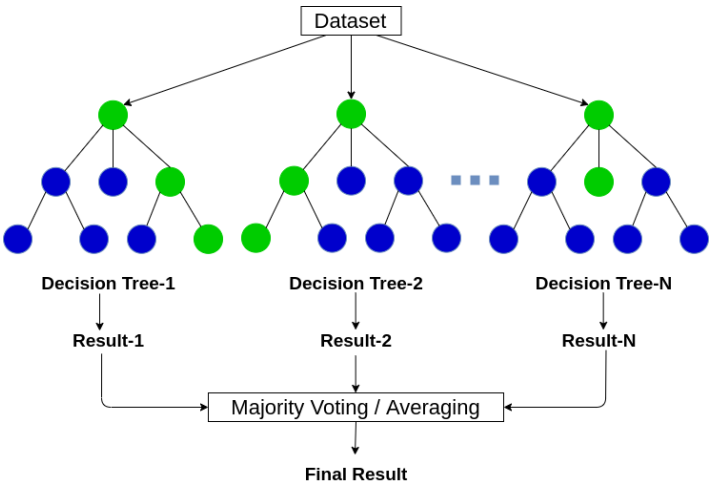


Figure 3: The structure of a random forest [29]

5.1.3 (Gradient) Boosting

The creation of a random forest model can happen in a parallelized way in terms of tree building as there is no connection between their structures in any way. This guarantees that these components remain uncorrelated provided that the bootstrapped datasets and the sampled attribute sets remain randomized. However, we could also consider the option of generating the trees iteratively with each tree increasing the performance of our model. This approach is called boosting. Since my work solely focused on tree-based algorithms, I have decided to explain these concepts using trees as base estimators, however trees can be substituted by all types of weak learners.

To explain the underlying concepts, I will introduce an early implementation of it in the form of Adaboost, which was proposed in 1999 by Yoav Freund and Robert Schapire [30]. Their solution assigns a set of weights to the training data and after every iteration, those weights that belong to incorrectly classified examples are increased as a function of that iteration's loss value. Freund and Schapire came up with the following weight function:

$$D_{(t+1)} = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha t}, & if\ h_t(x_i) = y_i \\ e^{\alpha t}, & if\ h_t(x_i) \neq y_i \end{cases}$$

, where x_i represents the i^{th} independent variables, y_i represents the i^{th} label in the dataset, D(t) represents the distribution of weights in the t^{th} iteration and h(t) the prediction function in the t^{th} iteration. α, the error function and Z_t, the normalization factor can be chosen as parameters.

As the formula suggests, the next model will have to pay greater attention to the misclassified samples and those are specifically the ones that would be otherwise harder to find.

This algorithm has held up relatively well even up until today, however gradient boosting and more specifically extreme gradient boosting (XGBoost), created by Tianqi Chen and Carlos Guestrin [31], seems to have become more popular recently. This surge in acceptance can be seen in the fact that most data science competitions have awarded XGBoost-based solutions more than any other. The main difference compared to simple tree boosting methods like Adaboost is that XGBoost uses regression trees instead of decision trees. These structures have basically the same properties, but regression trees contain continuous values as opposed to decision trees' discrete ones. This change makes it possible to create and calculate aggregate scores of trees and make them work together even more closely than Adaboost does. After every iteration we can calculate a pseudo residual value for the difference between the predictions and their expected labels. When creating a new tree, it is this value that we use to group our training set by and use it as target variable. This tree then will output a value that when added up to the sum of the previous predictions, will adjust them closer to their expected counterparts.

5.1.4 Neural networks

Artificial neural networks attempt to mimic the way our brains and other neural networks in nature function: these systems are made up of neurons and layers of neurons that interact with each other. A neural network must contain an input and an output layer and usually at least one hidden layer. Each neuron is assigned a function and each connection between the neurons is given a weight, then when a new input arrives, having applied their functions, the neuron layers will forward their output values to the next layer combined with the pre-determined weights and usually extended

by a bias value. In the classification tasks, provided that the network has been trained well, the output layer will contain a set of neurons, out of which only one has a mathematically significant value, while all the other ones are close to zero. When training a neural network, it is the weights and biases that get refined as more and more samples are shown.

I spent a great amount of my time researching relevant literature and trying to build a convolutional neural network (which is a subset of neural networks), that would perform well on my dataset, however in the end it just seemed like it was not meant to be. The best evaluation scores did not even come close to the tree-based classification metrics.

5.2 Evaluation metrics

Maximizing model performance has been mentioned several times, however I have not yet defined an exact way to judge, whether my solution actually is up to an adequate standard. The task is a binary classification problem and as such we need to look at functions that could potentially evaluate our results. A very easy, but in my opinion just as lazy choice would be to consider the accuracy score: I don't believe in this metric due to the skewed nature of my data. Having labeled the dangerousness of each action, I have gotten 26828 dangerous and 293112 non-dangerous samples and we can clearly see a ten-fold difference between the amounts. This would mean that if all our model did was guess 0 (as in not dangerous), it would reach an accuracy of around 0.9. In my opinion, it is more important to find the dangerous samples even at the expense of mislabeling some dangerous ones, which might even in turn lower our accuracy and precision. A high recall (sensitivity) value would mean just that: it measures the proportion of true positives and the sum of true positives and false negatives. This solution would fall into a trap on the other end of the spectrum: predicting that every event is dangerous would lead to 0 false negatives and as such we would get a recall value of 1. F1 score seems to tackle these sorts of issues, it is defined the following way:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2} * (FP + FN)}$$

As we can see, it incorporates both recall and precision through using the harmonic mean of these two scores.

Now we can start determining expectations for our model. In Figure 8, I have calculated F1 values for some edge cases for a dataset where there are n and 10 * n samples for the two classes:

Case	TP	FP	TN	FN	Precision	Recall	F1-score
Only P	n	10 * n	0 * n	0 * n	0.09	1	0.1651
½ P, ½ N	½ * n	5 * n	5 * n	½ * n	0.09	0.5	0.1525
$\frac{1}{11}$P, $\frac{10}{11}$ N	$\frac{1}{11} * n$	$\frac{10}{11} * n$	$\frac{100}{11} * n$	$\frac{10}{11} * n$	0.09	0.09	0.09
Only N	0 * n	0 * n	10 * n	n	0	0	0

Figure 8: F1-scores for randomized predictions

The first row shows a situation, in which recall is maximized, because we have recognized every positive sample, however it obviously happens at the cost of mislabeling all negative values. The second row shows the results we would get if we were to choose our predictions completely randomly. The third row represents a model which randomizes its evaluation, but also recognizes the class imbalance. And the fourth row would have the highest accuracy of all, but obviously would score the worst on our metric. This means that the bare minimum expectation for us would be to improve on the 0.2 threshold, however even then I thought that a score like that was way beyond achievable and as such I set out to at least reach an F1-score of 0.5.

5.3 Tools

As mentioned earlier, I have chosen to implement the algorithms in Python. Three libraries have helped me tremendously during my work: Scikit-learn [32], Xgboost [33] and Pytorch [34].

5.3.1 Scikit-learn

Also known as sklearn, Scikit-learn provides a great Machine Learning framework for data scientists with a plethora of built-in methods and classes. The library provides easily customizable machine learning algorithms, such as decision tree,

random forest and Adaboost, it has tools for the stages just before model creation, like splitting up the data into training and test sets or normalizing its values with scalers, such as a minmax or a standard scaler class and it can even maximize model performance with feature selection and grid search algorithms.

5.3.2 Xgboost

As its name suggests, this library helped me implement a working XGBoost model. While sklearn's strength lies in its uncountably many modeling, preprocessing, and evaluation utilities, xgboost is specialized for one certain use, and as such provides us with a huge set of options to customize it. The tool allows us to create an XGBoost model with different kinds of boosting algorithms, makes it possible to run it on our GPU while being able to process dozens of hyperparameters. It is also compatible with the model interface of sklearn, which ended up being important for me, because this way I could generalize my code and not have to create different methods for sklearn-based and xgboost-based solutions and as it turned out, a parameter that did not seem to work (namely pos_scale_weight) could be more or less substituted with an alternative from sklearn.

5.3.3 Pytorch

Pytorch along with Tensorflow are the most popular deep learning frameworks currently. As an open-source project, its development is heavily based on community insights, however Facebook's AI Research lab (FAIR) are also continually working on it. Tensorflow is owned by Google and as such the competition between the two corporations likely makes the development even more intense. A great feature of both of these frameworks is the ability to train and deploy models using GPUs, which can dramatically decrease the time needed for training. Unfortunately, this option is only available for Nvidia GPUs and as such I personally could not run it on my own computer. I did not want to let this hiccup make me wait significantly more for each training and test session, so I started looking up some alternative solutions. As it turns out, Microsoft have also started their own Deep learning venture, and as luck would have it, their solution, DirectML [35] allows us to move pytorch models to our GPU. The implementation was rather straightforward and as a result, a training epoch of 1497 samples with a batch size of 64 lasted 20 seconds instead of 2 minutes and 23 seconds, which means, I managed to speed up the process by a factor of 7.

5.4 Implementation with tree-based classification

Even before starting the implementation, I had to concede a crucial fact: since the danger values were determined as a function of location, there was a real possibility of that somehow impacting my model. To counteract this problem, I made two important measures, first I split my training set into three categories: actions in the defensive phase (at most 40 meters away from the attacking team's goal), actions in the midfield (between 40 and 75 meters away from the attacking team's goal), and actions in the attacking phase (at least 75 meters away from the attacking team's goal), my expectation was that this change would negate any threat of bias in my model, while the other change might have had less significant effects: I removed the x and y coordinates of the events from my training data set, which meant that only the players' attributes remained (with the time elapsed and action type).

I finally had a cleaned and ready-to-go dataset that contained more than 100 attributes. It was time to create a pipeline that would output a trained machine learning model. Pandas, numpy and sklearn offer plenty of tools for this stage to be relatively easily manageable. The pipeline is made up of the following steps:

- Reading the parquet file and turning it into a pandas DataFrame
- Selecting the relevant columns from the DataFrame
- Dropping those rows that still contain Nan values
- Dividing the data up into 3 DataFrames (attacking phase, midfield, defending phase)
- For each DataFrame:
    - Creating a model object with the corresponding hyperparameters
    - Splitting it up into training and test datasets
    - If the model has been trained and saved (using pickle):
        - Loading the trained model
    - Else:
        - Training the model on the training set
        - Saving (pickling) the trained model
    - Displaying the most relevant scores and a confusion matrix

A seemingly significant oversight on my part could be the fact that I did not normalize the values stored in the DataFrames, as that is a step, that usually is taken

when training a model. Most distance-based classification methods would give higher priority to higher non-scaled values, however when it comes to trees, this aspect is not significant, if relevant at all.

Another important aspect to note is that during the training phase, the algorithm had to account for the imbalanced nature of my dataset. Sklearn provides two similar solutions for this sort of problem: we can either assign weights to classes when defining the model object, or individually assign weights to each sample. Due to a sort of incompatibility between the xgboost and sklearn models, I had to come up with a hybrid solution: xgboost did not allow class weights as parameter and as such I created a function, that assigned a parametrizable value to samples belonging to either class.

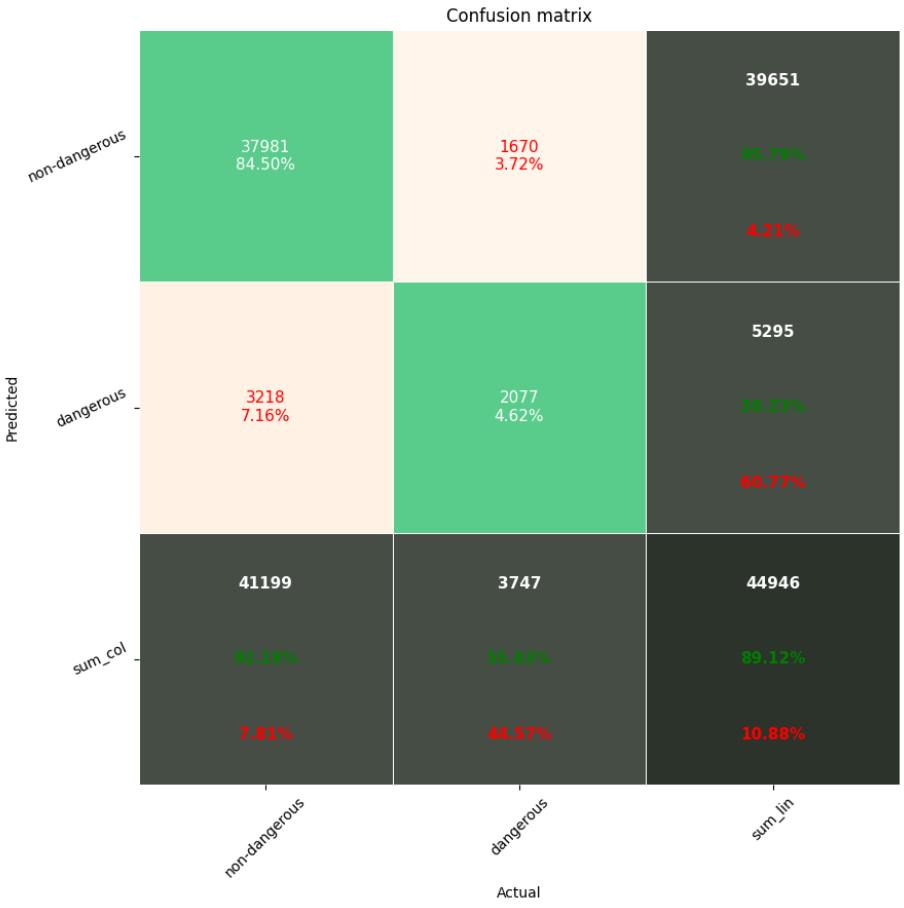


Figure 9: Initial results

As Figure 9 suggests, my initial results seemed promising with a recall score of 0.5543, and a precision score of 0.3923, the F1-score ended up at 0.46. This value did not live up to my original expectations, but there was still work to be done. This confusion matrix was created based on the outputs of an XGBoost model, which

seemed to achieve slightly better F1-scores than a random forest. As a next step, I started to finetune the models' hyperparameters, which I hoped would greatly increase model performance. After reading through some recommendations in the official xgboost documentation [33], I decided to mainly focus on the following hyperparameters:

- Booster (Final value: 'dart')

  I had the choice of using the basic XGBoost algorithm or trying a slightly upgraded version with randomized dropping of trees called 'dart' (Dropouts meet Multiple Additive Regression Trees. This extra feature reduces the possibility of overfitting

- Learning rate (Final value: 0.2)

  This parameter basically determines how much we want to weigh the loss values after each iteration.

- Subsampling ratio (Final value: 0.6)

  What proportion of samples is provided to each tree. Setting a lower value can help prevent overfitting.

- Column sampling ratio at each node (Final value: 0.6)

  What proportion of attributes is provided to each node. Setting a lower value can help prevent overfitting.

- Max depth for trees (Final value: 90)

  How deep we want to let trees grow? A higher value might cause higher accuracy, but we need beware of overfitting and resource constraints.

- Sample weights (Dangerous: 0.975, Non-dangerous: 0.025)

  As discussed above, this value accounts for class imbalance.

Having found out possible values for each hyperparameter, it was time to implement a grid search algorithm, that explores the performances of every combination of hyperparameters. Sklearn has an implementation of this method, however the slight incompatibility between sklearn and xgboost meant, that I had to create one for myself. After determining the final values, I built my model and got the results shown in Figure 9. As we can see, this model achieved a recall score of 0.59 and a precision of 0.68, we can calculate then, that it reached an F1-score of 0.63, which far exceeded my initial expectation of 0.5.

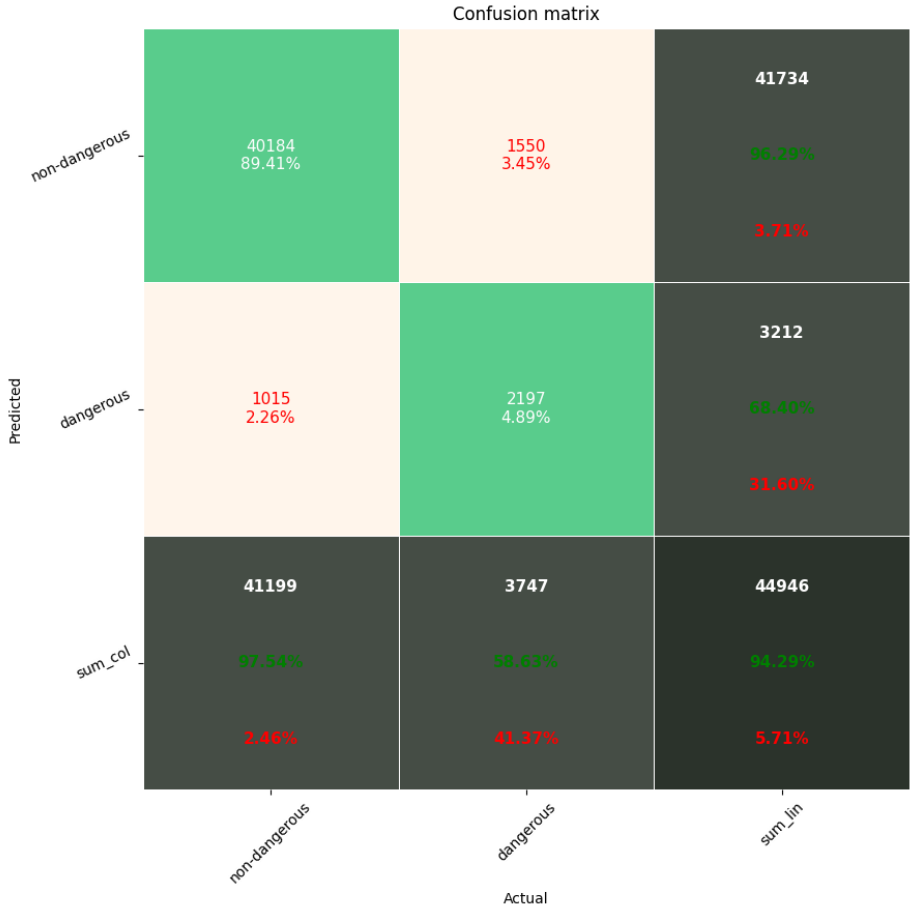


Figure 9: Final results

6. Evaluation

In this section, I will solely be focusing on the tree-based classification results as I have discussed earlier, in my judgement, the neural network, while certainly an intriguing option, did not live up to my expectations.

6.1 Evaluating performance

It is relatively hard to directly compare my findings to other works, as most state-of-the-art solutions had lot more data to validate their findings on and most of them implemented regression-based algorithms as opposed to my binary classification task. However, I tried to find common ground and base my comparisons on metrics mentioned in other papers, that I had researched earlier.

6.1.1 Traditional scoring methods

As I have noted in Paragraph 5.3.1, my solution ended up reaching a recall score of 0.59, a precision of 0.68, which we can combine to end up with an F1-score of 0.63. For curiosity's sake, we can also calculate these values in terms of the non-dangerous samples. These scores are definitely not as significant, when it comes to evaluating my results, but to get a full picture, I believe that it is important, we learn these values too. If we consider the non-dangerous samples as the positive ones, we get a recall score of 0.9854, a precision of 0.9629, the harmonic mean of these two values is approximately 0.9691. Sklearn proposes taking either the average of the two F1-score, which would yield an overall F1-score of 0.8 or calculating the weighted average of these two values, which produces an even better result of 0.94. I am well aware of the fact, that these values are inherently biased, and do not focus on the primary goal of the task at hand, however, I believe that these scores are required to see the whole picture.

6.1.2 Correlation with goals

Van Roy et al. [36] propose in their paper the correlation of their model outputs to baseline metrics, like goals scored. They were fortunate enough to have a larger database at their disposal and as such were able to correlate their values to players' goals scored per 90 minutes. To overcome the lack of data I calculated the correlation coefficient between the number of situations leading to dangerous situations for each

team and their goals scored per 90 minutes. The Expected Threat (xT) model achieved a correlation coefficient of 0.41, while the Valuing Actions by Estimating Probabilities (VAEP) model reached a moderate score of 0.26, whereas my solution managed to achieve a correlation coefficient of 0.63. This high level of correlation compared to the state-of-the art solutions is great to see, however players' performances tend to show a greater variance than teams' and as such, should be treated accordingly.

6.1.3 Features processed

VAEP [10] uses 151 different attributes to base its predictions on, these include categorical data, such as the action type that occurred with the event, location data, such as the x and y coordinates, or the distance and angle to goal and context features, like the current goal difference. My set of features is made up of 163 attributes, however those contain mostly the same kind of data which is related to player movement. This results in a way more specialized model that does not need a huge variance of data, which makes it more accessible, however this might also decrease performance.

6.1.4 Comparing with traditional player rating metrics

In terms of concrete offensive scoring metrics, it is goals and assists, that are usually tallied for players. Decroos et al. [9] explain in their work, that what could set a model like ours apart from these is the ability to recognize other players that do not necessarily excel in the area of finishing or key passes. The top 10 players with the most Contributions to Danger (CtD) according to my metric have been:

1) Sadio Mané (Winger) - 22.75 CtD per game
2) Taison Barcellos Freda (Midfielder) - 19.33 CtD per game
3) Diego Perotti (Winger) - 19.0 CtD per game
4) Yacine Brahimi (Winger) - 18.5 CtD per game
5) Kevin de Bruyne (Midfielder) - 18.0 CtD per game
6) Mohamed Salah (Winger) - 17.75 CtD per game
7) Ilkay Gündogan (Midfielder) - 16.33 CtD per game
8) Leroy Sane (Winger) - 16.0 dangerous CtD per game
9) Kyle Walker (Defender) - 15.33 CtD per game
10) Quincy Anton Promes (Winger) - 15.33 CtD per game

As we can see, outside of Mané, de Bruyne and Salah, most players on this list are not known for their goal scoring or assisting prowess, what is more, we can find players like Walker and Gündogan, who were not even playing in attacking positions

then (Gündogan has since been advanced into a more offensive role for the 2020/21 season). It is important to note, that most of the players on this list are lauded as important parts of their respective teams, which further strengthens the credibility of my predictions.

6.2 Analyzing model properties – Feature importances

Exploring the way our model works might allow us to draw conclusions that we can then project to real life.

Xgboost provides easy access to the relative weights of features in a model. Looking at the list of feature importances, part of which is displayed in Figure 10, it has become quite clear to me, that there is still a bias towards the x coordinates of players, mainly those defenders from one team and attackers from the other that are the closest to the goal. It should not come as a surprise however, that for example the positioning of the goalkeeper (who should be the closest defender to the goal at all times) does not play a significant role when evaluating a situation. Strangely enough, the y coordinates do not seem to have a huge effect on the outcome. Among the high-ranking attributes, we can also find the elapsed time, which means that my hypothesis of fatigue impacting situations has turned out to be true.

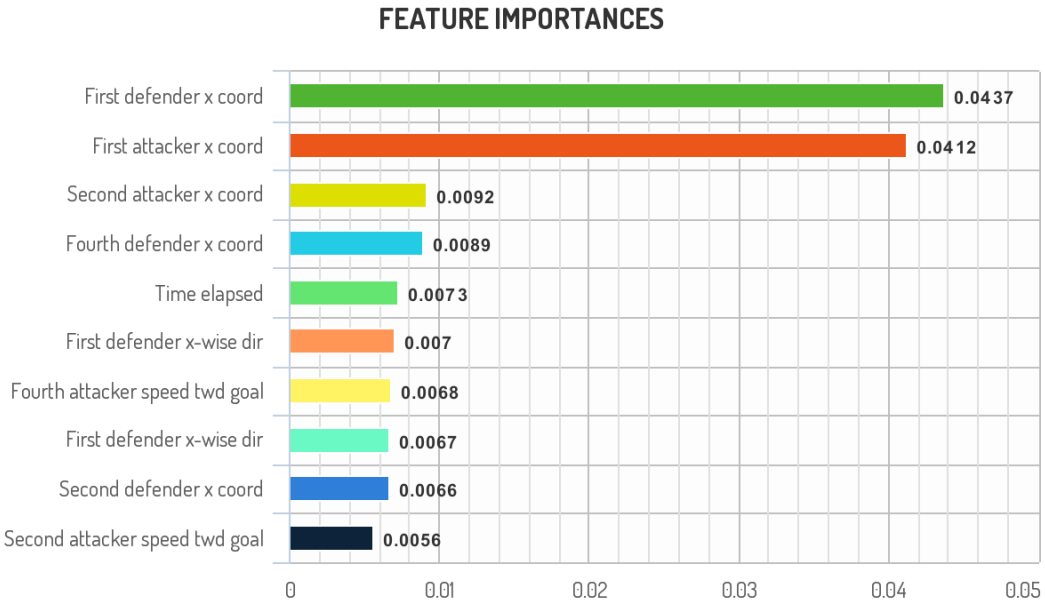


Figure 10: The 10 most important features

Due to the way categorical attributes are handled in xgboost, it was not possible to derive a single importance value for the event type, because for each type, I had to

generate a new column containing binary values (either a situation belonged to an event type or not). What I concluded from these separate feature importances for each column was the fact that those mostly helped find the negative samples, because the highest weights were assigned to losses of ball and other sorts of mistakes.

6.3 Future work

Even at the very beginning of my work, I was looking very optimistically forward to the opportunity of building working convolutional neural networks. We can see in our everyday lives that more and more solutions incorporate such models one way or another, and I even found papers about deep learning in soccer analytics [14] [37] and saw very promising results. As such my enthusiasm and optimism got the better of me and I ended up spending a disproportionate amount of time trying to implement a model that yields acceptable results. I ended up not succeeding and having to move on to other areas due to the lack of time, however I still strongly believe that projecting a situation into an image format the way that as an example Fernández & Bornn [37] did is a very interesting option that is worth looking into even more.

I am also strongly of the opinion that an even richer data set could improve my model's results tremendously. It is not unheard of to train a model on a season's worth of matches in a league, which adds up to around 400 games, whereas due to the splitting required for test samples, I had to make do with around 70 matches worth of samples. This sort of 5-fold increase in training data size could certainly help my work. This seems a lot more unrealistic however, due to the nature in which my requests were rejected at the start of the semester.

While there is certainly relevant use of this sort of data, I am of the opinion, that finding a way to convert my model into a regression model instead of the current binary classifier would make its findings even more useful. This could be achieved by introducing Expected Goals (xG) as a new attribute, and instead of labeling each possession with a 0 or 1, we could examine the highest xG value the possession has reached and assign that to the whole possesion. Another solution would be to train a logistic regression model, which would output a value between 0 and 1.

7. Deployment

With a readily available model, there is plenty of options to analyze and visualize matches and situations. I have decided to create two separate tools that could potentially help both managers preparing for opponents or reviewing their own games, and also provide fans with insight into how some teams try to attack and what formations they aim to take. My first utility is able to generate and display a heat map of where each CtD has occurred for each team according to my model, while the other one allows its users to recreate any sort of situation on the pitch and have my model evaluate whether that will end up in a dangerous action.

As it normally goes with web development projects, I divided the task and code base into two separate units, front-end and back-end.

7.1 Front-end development

A front-end developer has to make sure that the users have a comfortable experience interacting with the website. To achieve this, it is crucial to design and program the website's appearance properly, while finding edge cases that the visitors could exploit or find inconvenient. In short, they are responsible for the workings of the User Interface. [38]

7.1.1 HTML, CSS, JS

While it is certainly possible to create web interfaces in Java, Python or any other popular programming language, JS (JavaScript) usually in combination with an additional framework is the main choice when it comes to front-end development. JS helps create the logic behind our site with HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) providing a relatively easy way to display it the way we want to. I did not have any particular reason not to proceed with this set-up, and as such could decide whether I would be able to create the platform using 'Vanilla JS', which just means without the use of any frameworks, or if I needed the help that one could offer.

7.1.2 Vue.js

Last year, while completing my internship, I had the chance to try out and work with a framework, called Vue.js [39]. Having had a very pleasant experience using it, due to its simplicity, lack of further knowledge required and great level of customizability, I decided, that I would utilize it for this project as well. In a similar fashion to other frameworks, a great advantage of Vue over simple JS is the fact that new custom components can be created very easily. As such, instead of having to rely on more convoluted coding, I was able to create a web page with the following custom components:

<Base> - This component serves the purpose of displaying the custom created components

<HeadLine> - This component represents the header of my page

<TeamSelector (conditional)> - This component helps choose which team's heatmaps we want to display

<HeatMap (conditional)> - This shows the chosen team's pre-generated heatmap

<Predictor (conditional)> - This component waits for the user to set the correct parameters and then after communication with the back-end returns the evaluation output.

A noteable feature of Vue.js is conditional rendering (attribute: v-if), which allows the site to only display certain components when instructed. In this case either the TeamSelector and HeatMap components are shown or the Predictor. A similar functionality is list rendering (attribute: v-for), which is a very easy way to create a list of identical sections, this helped me list the team names and create sliders for the classification parameters.

7.1.3 TeamSelector & HeatMap

In terms of front-end coding needed, creating these components was rather straightforward. Having list rendering at my disposal meant that for the TeamSelector it was only some stylistic choices that I had to make apart from obviously binding the texts to the actual team names, so that when they get clicked on, a new request is sent immediately.

Only having to deal with 11 teams meant that I could just generate the heatmaps prior to deployment and place them into the public folder. I have chosen to create those images using Python and its matplotlib [40] and seaborn [41] libraries with the help of a

learning resource specifically created for sports analysts, FC Python [42]. After each click on TeamSelector, the HeatMap component changes its image source attribute.

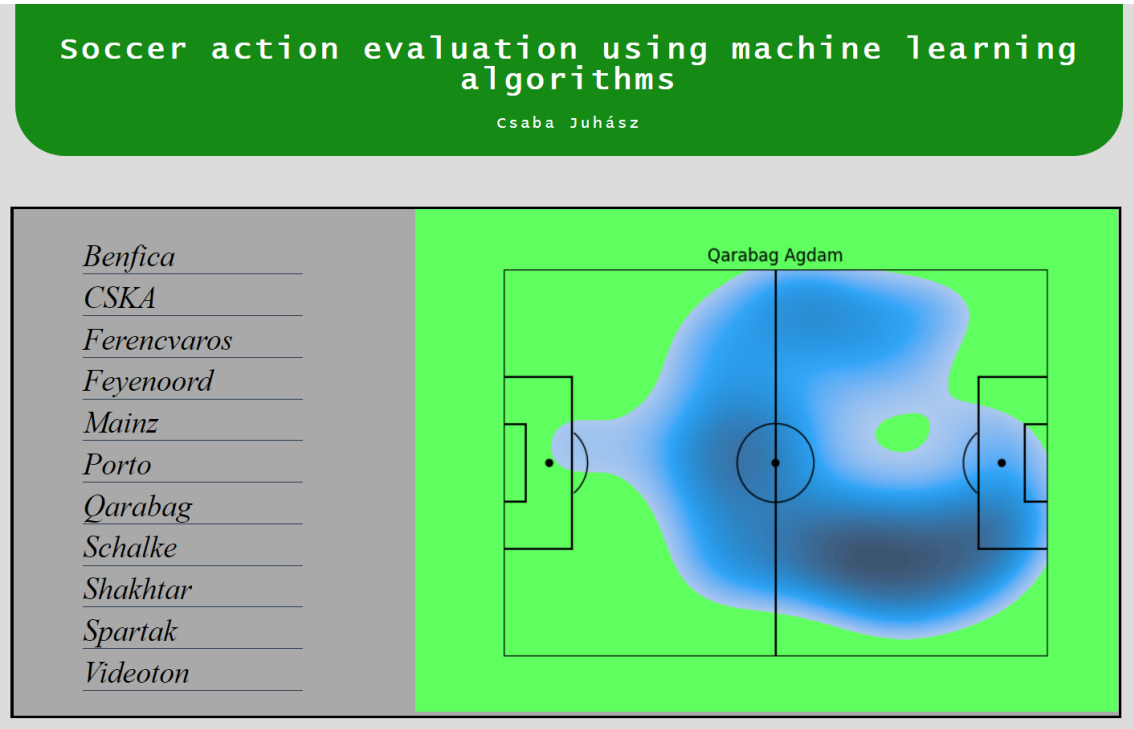


Figure 11: Qarabaq's CtD heatmap displayed on my web page

As Figure 11 clearly shows, there is actually a lot that we can learn about teams from these images. In the case of Qarabag, it is the fact that they mainly rely on their right flank when attacking even though it is their left-sided center back who plays in a more ball-playing role as it seems. An opponent could also expose Qarabag's clear weaknesses on the left wing and left half-space.

7.1.4 Predictor

This component required the use of canvas, which is an HTML element that can be used to draw graphics via scripting. Having had no experience using any tools similar to this one, it took me some time to get to grips with it.

As a first step, I had to draw a football pitch manually using its line and arc creation features. With that done, I added an event listener to it, that handles clicks on the pitch. The first 11 clicks locate the defending players (colored red), the second 11 locate the attacking players (colored blue), see Figures 12 and 13. We need to save these values in a list sorted by the x coordinate and normalize them to the pitch dimensions of 105x68.

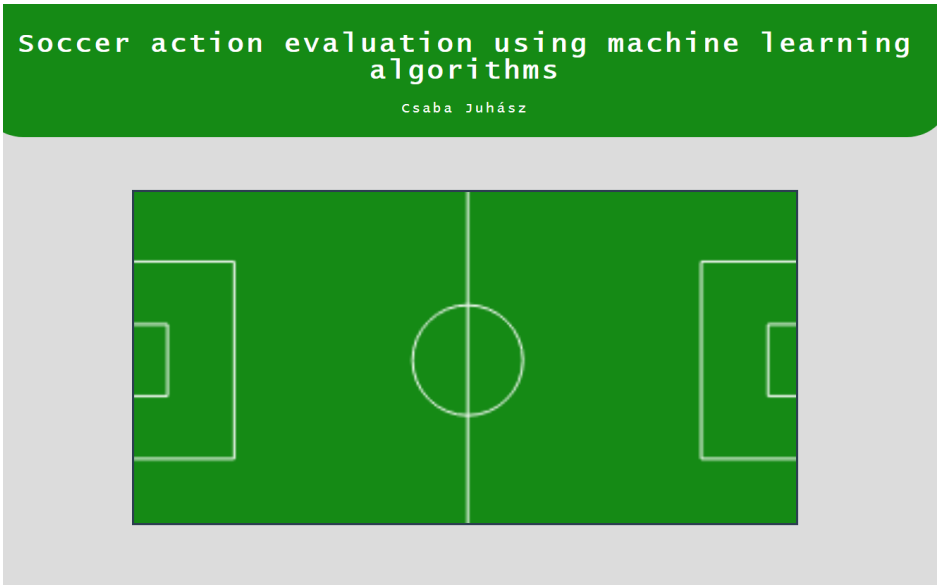


Figure 12: An empty pitch drawn in canvas

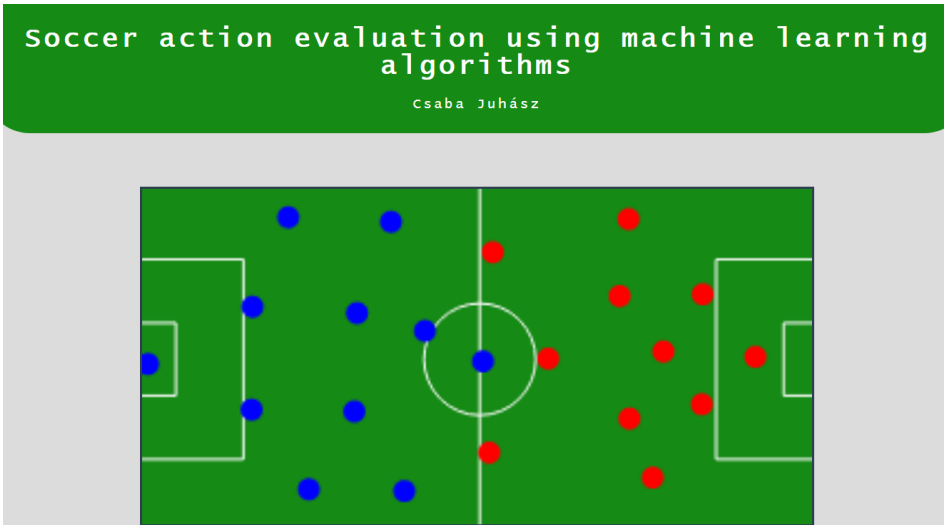


Figure 13: A canvas with player positions selected

Next, we create the sliders that set the additional attributes, such as the acceleration, speed and the rate of the x and y directional movements for the 5 closest players to the goal on each team. In order for the web page not to seem too congested, I decided to not include sliders for the acceleration and speed towards goal as those values can be derived from the previous 6 and I also decided that I would use a bit weaker version of my model that only expects the 5-5 closest players to the goal and their attributes. I also implemented conditional rendering for the sliders so that they don’t show up before placing the 22 players. These sliders are shown in Figure 14.

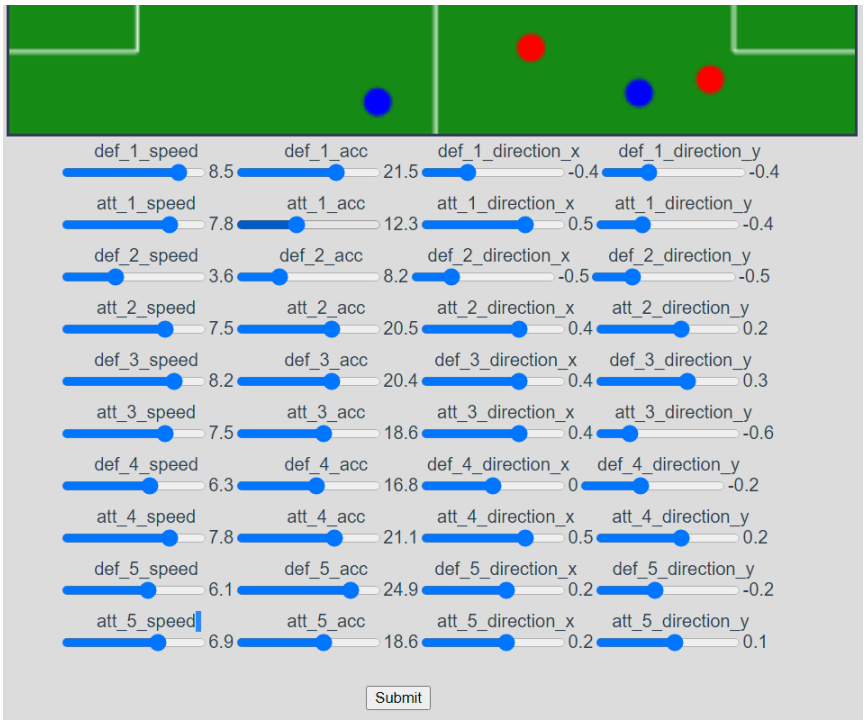
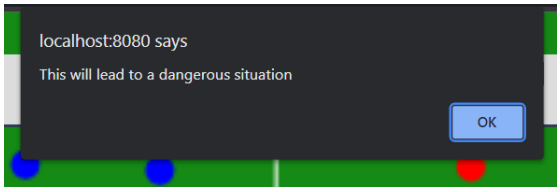
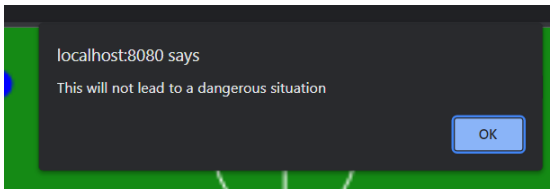


Figure 14: The sliders for the model parameters

After setting the correct values and clicking on the ‘Submit’ button, an HTTP GET request is sent using axios [43] to the back-end with the established parameters. Then on receiving a response, we display a message depending on the prediction of our model, shown in Figures 15 and 16.





Figures 15 & 16: Messages after positive and negative predictions

7.2 Back-end development

While front-end developers deal with what users are able to interact with, back-end development focuses on server-side processes that occur “behind the scenes”. These tasks include database and resource management, communication with external APIs and also integrating modules, such as a machine learning model. There is no single choice for back-end development when it comes to programming language, Python, Java, PHP and even JavaScript are completely legitimate options. As the back-end side did not contain any specific requirement, that would depend significantly on which

programming language I choose, I decided that for simplicity's sake I would create my back-end in Python.

Using the Flask [44] framework, I created an endpoint that would monitor HTTP GET requests on the 8000 port on localhost (localhost:8000/predict). I configured this end point in a way that if any parameter for the model is not included in the URI, the request gets rejected. When a valid request arrives, I calculate the speed_towards_goal and acceleration_towards goal values, sort the attributes in the correct order, load my model from a pickle file and use it to evaluate the incoming parameters, the model output is then returned to the client side.

8. Summary

During my work, I got the opportunity to learn a great deal about the world of data science and sports analytics, while also managing a data science project from start to finish.

I read up on several papers that dealt with machine learning solutions and algorithms, which helped me use and implement them in a more reasoned manner instead of basing my results on trial-and-error. I would like to highlight the XGBoost algorithm in particular, that I had not even heard of prior to my work, however by getting to know its core concepts, I managed to implement it successfully. The same can be said about the sports analytics side of my project: I already had some knowledge coming into this semester, however the papers by Decroos and Fernández were really eye-opening.

I was fortunate enough to have had some experience dealing with smaller data sports analytics, because my Project Laboratory was already spent dealing with this area. However up until now, I have not had to go through the data cleansing and data processing stage in such depths. It initially sounded unbelievable, but I need to concede the fact that the most time was spent completing this step of my work with some great difficulties along the way. In the end however, I ended up creating a custom dataset that did not have a significant number of missing values, and in my opinion its features were all relevant to the task at hand.

The model creation phase of my work, in particular when trying to build a working neural network, started off in a difficult fashion, as initially the results did not seem to suggest much promise. A rather significant oversight on my part was the fact that for a long while I did not find any substitutes for the class weight parameter in the XGBoost object, and it was after I managed to overcome this issue, that the more encouraging results started coming in. After that with the help of the grid search and other utilities, I was able to find an adequate combination of parameters for my model.

The F1-score of 0.63 could be improved through ways that I have proposed in Paragraph 6, however I am generally content with this outcome. As mentioned in my thesis, there was no way to directly compare my results to other existing solutions. I

tried to find ways to match CtD at least partially to VAEP and thankfully they seemed to perform at least on the same level.

I firmly hold the view that accurate models and results are worth nothing without a way of understanding their underlying workings. That is why I found it important to not only create this model, but also show how it operates and create concrete examples for its functionality. My web page -while certainly not the prettiest- accomplishes just that: both the heatmap data and the predictor module clearly show my outputs and we can examine them through different lenses when we actually see them.

9. Acknowledgements

I would like to first and foremost thank my consultant, László Toka for the extraordinary support he provided me with. During our weekly meetings, he kept on suggesting great ideas and solutions for my problems, which I will be eternally grateful for.

My work would not have been as successful without the data provided by Xfb. Analytics and as such I would like to highlight their contributions. Márton Wernigg and Attila Csernus, who work for the company have even spent a great amount of time consulting with me and proposing their own ideas.

10. References

"What is CRISP DM?," [Online]. Available: https://www.datascience-pm.com/crisp-dm-2/. [Accessed 3 Dec 2021].

[1]

"IBM: Data Preparation Overview," [Online]. Available: https://www.ibm.com/docs/en/spss-modeler/SaaS?topic=preparation-data-overview. [Accessed 3 Dec 2021].

[2]

L. Steinberg, "CHANGING THE GAME: The Rise of Sports Analytics," 18 Aug 2015. [Online]. Available: https://www.forbes.com/sites/leighsteinberg/2015/08/18/changing-the-game-the-rise-of-sports-analytics/?sh=549190cf4c1f. [Accessed 10 Oct 2021].

[3]

M. Lewis, Moneyball: The Art of Winning an Unfair Game, W. W. Norton & Company, 2003.

[4]

J. Tippett, The Expected Goals Philosophy: A Game-Changing Way of Analysing Football, Independently published, 2019.

[5]

C. Biermann, Football Hackers: The Science and Art of a Data Revolution, Blink Publishing, 2019.

[6]

D. Cervone, A. D'Amour and L. Bornn, "POINTWISE: Predicting Points and Valuing Decisions in Real Time with NBA Optical Tracking Data," in *MIT SLOAN Sports Analyitics Conference, 2014*, 2014.

[7]

KU Leuven, "Our publications," [Online]. Available: https://dtai.cs.kuleuven.be/sports/publications. [Accessed 10 Oct 2021].

[8]

T. Decroos, L. Bransen, J. Van Haaren and J. Davis, "Actions Speak Louder Than Goals: Valuing Player Actions in Soccer," in *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[9]

T. Decroos, L. Branse, J. Van Haaren and J. Davis, "VAEP: An Objective

10] Approach to Valuing On-the-Ball Actions in Soccer.," in *International Joint Conferences on Artificial Intelligence Organization, 2020*, 2020.

11] K. Singh, "Introducing Expected Threat (xT)," 2019. [Online]. Available: https://karun.in/blog/expected-threat.html. [Accessed 20 Oct 2021].

12] J. Fernández, L. Bornn and D. Cervone, "Decomposing the Immeasurable Sport: A deep learning expected possession value framework for soccer," in *MIT SLOAN Sports Analytics Conference, 2019*, 2019.

13] M. Van Roy, P. Robberechts, W.-C. Yang and J. Davis, "Learning a Markov Model for Evaluating Soccer Decision Making," in *Reinforcement Learning for Real Life Workshop, 2021*, 2021.

14] G. Liu, Y. Luo, O. Schulte and T. Kharra, "Deep soccer analytics: Learning an action-value function for evaluating soccer players," *Data Mining and Knowledge Discovery,* September 2020.

15] R. Smith, "How Arsenal and Arsène Wenger Bought Into Analytics," 3 Feb 2017. [Online]. Available: https://www.nytimes.com/2017/02/03/sports/soccer/arsenal-arsene-wenger-analytics.html. [Accessed 26 Nov 2021].

16] "Opta Data by Stats Perform," [Online]. Available: https://www.statsperform.com/opta/. [Accessed 26 Nov 2021].

17] InStat, "InStat Sport," [Online]. Available: https://instatsport.com/. [Accessed 26 Nov 2021].

18] Transfermarkt, "Transfermarkt," [Online]. Available: https://www.transfermarkt.com/. [Accessed 26 Nov 2021].

19] WhoScored, "WhoScored," [Online]. Available: https://www.whoscored.com/. [Accessed 26 Nov 2021].

20] K. Stuart, "Why clubs are using Football Manager as a real-life scouting tool," 12 Aug 2014. [Online]. Available: https://www.theguardian.com/technology/2014/aug/12/why-clubs-football-manager-scouting-tool. [Accessed 26 Nov 2021].

"The market values of teams in the 2017/18 season in the Champions
21] League," [Online]. Available: https://www.transfermarkt.com/uefa-champions-league/teilnehmer/pokalwettbewerb/CL/saison_id/2017. [Accessed 27 Nov 2021].

"The market values of teams in the 2017/18 season in the German
22] Bundesliga," [Online]. Available: https://www.transfermarkt.com/1-bundesliga/startseite/wettbewerb/L1/saison_id/2017. [Accessed 27 Nov 2021].

"The market values of teams in the 2017/18 season in the Nemzeti
23] Bajnokság I," [Online]. Available: https://www.transfermarkt.com/nemzeti-bajnoksag/startseite/wettbewerb/UNG1/plus/?saison_id=2017. [Accessed 27 Nov 2021].

C. D. Costa, "Top Programming Languages for Data Science in 2020," 24
24] Aug 2020. [Online]. Available: https://towardsdatascience.com/top-programming-languages-for-data-science-in-2020-3425d756e2a7. [Accessed 28 Nov 2021].

"About pandas," [Online]. Available: https://pandas.pydata.org/about/.
25] [Accessed 28 Nov 2021].

S. Asiri, "Machine Learning Classifiers," 11 Jun 2018. [Online]. Available:
26] https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623#. [Accessed 20 Oct 2021].

N. Bassiliades, "A simple decision tree classifier with 4 features," Nov
27] 2019. [Online]. Available: https://www.researchgate.net/figure/A-simple-decision-tree-classifier-with-4-features_fig2_337413360. [Accessed 20 Oct 2021].

A. Sharma, "Decision Tree vs. Random Forest – Which Algorithm Should
28] you Use?," 12 May 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/. [Accessed 20 Oct 2021].

T. Yiu, "Understanding Random Forest," 12 Jun 2019. [Online]. Available:
29] https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Accessed 20 Oct 2021].

Y. Freund and R. Schaphire, "A Short Introduction to Boosting," 1999.

30]

T. Chen and G. Carlos, "XGBoost: A Scalable Tree Boosting System," in
31] *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Scikit-learn, "Scikit-learn," [Online]. Available: https://scikit-
32] learn.org/stable/. [Accessed 3 12 2021].

xgboost developers, "XGBoost Documentation," [Online]. Available:
33] https://xgboost.readthedocs.io/en/stable/. [Accessed 3 Dec 2021].

Pytorch developers, "Pytorch Documentation," [Online]. Available:
34] https://pytorch.org/. [Accessed 3 Dec 2021].

Microsoft, "Introduction to DirectML," Microsoft, 13 Oct 2021. [Online].
35] Available: https://docs.microsoft.com/en-us/windows/ai/directml/dml-intro. [Accessed 5 Dec 2021].

M. Van Roy, P. Robberechts, T. Decroos and J. Davis, "Valuing On-the-
36] Ball Actions in Soccer: A Critical Comparison of xT and VAEP," in *Proceedings of the AAAI-20 Workshop on Artifical Intelligence in Team Sports; 2020*, 2020.

J. Fernández and L. Bornn, "SoccerMap: A Deep Learning Architecture for
37] Visually-Interpretable Analysis in Soccer," in *EMCL-PKDD 2020 conference proceedings, Applied Data Science Track*, 2020.

UC Berkeley Extension, "What Does a Front End Web Developer Do?,"
38] UC Berkeley Extension, 2021. [Online]. Available: https://bootcamp.berkeley.edu/resources/coding/learn-web-development/what-does-a-front-end-web-developer-do/. [Accessed 6 Dec 2021].

Vue.js developers, "Vue.js," Vue.js, [Online]. Available: https://vuejs.org/.
39] [Accessed 6 Dec 2021].

Matplotlib developers, "Matplotlib: Visualization with Python," [Online].
40] Available: https://matplotlib.org/. [Accessed 7 Dec 2021].

Seaborn developers, "seaborn: statistical data visualization," [Online].

41] Available: https://seaborn.pydata.org/. [Accessed 7 Dec 2021].

"Learn Python & Data Science With Football," [Online]. Available:
42] https://fcpython.com/. [Accessed 7 Dec 2021].

Axios developers, "axios," Github, [Online]. Available:
43] https://github.com/axios/axios. [Accessed 6 Dec 2021].

Flask developers, "Flask," Flask, [Online]. Available:
44] https://flask.palletsprojects.com/en/2.0.x/. [Accessed 6 Dec 2021].

E. Adeyemi-Abere, "Atlético Madrid – Liverpool: Majestic Mo Seals
45] Victory In Testing Trip To The Wanda (2-3)," 24 Oct 2021. [Online]. Available: https://betweentheposts.net/atletico-madrid-liverpool-majestic-mo-seals-victory-in-testing-trip-to-the-wanda-2-3/. [Accessed 10 2021].

T. Spicer, "Apache Parquet vs. CSV Files," 28 May 2019. [Online].
46] Available: https://dzone.com/articles/how-to-be-a-hero-with-powerful-parquet-google-and. [Accessed 28 Nov 2021].

Wyscout, "Wyscout," [Online]. Available: https://wyscout.com/. [Accessed
47] 26 Nov 2021].