Using Soccer Athlete GPS Monitoring Data to Visualize and Predict Features

Lars Hoel

Thesis submitted for the degree of
Master in Computer Science: Programming and System Architecture
60 credits

Department of Informatics
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2023

Using Soccer Athlete GPS Monitoring Data to Visualize and Predict Features

Lars Hoel

Using Soccer Athlete GPS Monitoring Data to Visualize and Predict Features

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Football is a globally popular sport with millions of players and fans engaging in the game across all levels of competition. As one of the world's most-watched sports, football demands constant improvements in data analysis tools. This master's thesis presents a comprehensive pipeline for feature extraction, data visualization, and injury prediction, utilizing GPS data collected from two Norwegian women's soccer teams. It outlines the development of a systematic process, commencing with preprocessing and feature extraction from raw GPS data, to facilitate subsequent analysis and model training. This process culminates in the creation of two distinct datasets - 'Session' and 'High Intensity Run' - which offer invaluable insights into player performance and physical attributes.

The study then delves into the creation of several visualization tools, utilizing a mix of the aforementioned datasets, raw data, subjective performance data, and match data. The resulting visualizations serve diverse purposes, providing insights into high-intensity runs, player positions, team heatmaps, and the relationships between subjective game performance, objective GPS metrics, and match data. These tools exhibit potential in assisting players, coaches, medical staff, researchers, and sports scientists in a multitude of scenarios, such as managing tactics, preparing for high-intensity periods, and evaluating player mindsets.

Lastly, the thesis explores injury prediction through the deployment of various machine learning models. After testing several models, including Logistic Regression, Decision Tree, xGBoost, LSTM, GRU, and ROCKET, the ROCKET model is found to outperform others for the given dataset, with precision of 0.4167 and recall of 0.4545 (TP:5, TN:2978, FP:6, FN:7). However, the model's performance is found lacking in consistently predicting injuries, thereby underscoring the need for continued research in this field. This study's comprehensive process and findings contribute significantly to enhancing our understanding of the application of GPS data in professional sports, while pinpointing areas for future investigation.

Acknowledgments

This thesis could not have been accomplished without the mentorship and assistance of numerous individuals who contributed in various ways to the development and completion of this study.

First and foremost, I wish to express my appreciation to my supervisors, Professor Pål Halvorsen and Dr. Cise Midoglu, for their guidance, astute observations, thorough recommendations, and unwavering support throughout this thesis project. Additionally, I am grateful to Steven Hicks and PHD student Matthias Boeker for their assistance at various stages of the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Soccer is the most popular sport in the world, both played and watched by millions of people from all over the world. As the sport continues to evolve, so does the technology used to capture and analyze various components of the game [75]. At the top level of the game, everything from player movements, to ball trajectories, to physiological data and performance metrics is tracked and analyzed to help stakeholders including players, coaches and medical staff make better, and more informed decisions about team lineup, effective injury prevention, physical form, etc. [24, 59, 75, 78]. All in all, the possibility for gathering larger and more diverse datasets opens up new avenues for teams to make data-driven decisions and it provides fans with more detailed insights into the game.

1.1 Motivation

The field of sports analytics experiences an uptick in popularity in recent years, with teams and organizations from all around the world turning to data-driven methods to enhance their performance and gain a competitive edge over the competition [75]. With the technological advances of the last couple of decades, sport analysts create large data and diverse datasets, from which they extract valuable insights and identify patterns and trends previously unavailable. This thesis focuses on extracting useful features from the SoccerMon dataset [59], containing data from a Global Positioning System (GPS) [24] and subjective wellness, performance and injury data, that includes data like fatigue, readiness, soreness, game performance and other self reported values, with the aim of providing valuable insight into performance, form and wellness of the athletes.

The use of GPS data in sports analytics becomes more and more popular over the recent years [3]. GPS data enables teams to track things like player movement, speed, heart rate, and distance covered. GPS data also gives a more detailed understanding of the physical demands of the sport [3].

Even though this thesis mostly focuses on the GPS data, the subjective wellness, performance and injury data is also used for analysis. The

inclusion of the subjective data provides additional insights into the factors that affect player performance.

The use of subjective feedback is a crucial method for monitoring an athlete's performance, as it allows coaches to track progress and identify areas for improvement [59]. However, the reliability of this data can vary, as it is based on subjective assessments of an athlete's well-being [49]. Prediction models and other analysis tools that rely on accurate wellness data may perform poorly, underscoring the need to develop accurate questions to elicit precise responses from athletes [88]. Despite these challenges, significant research is conducted in the field of subjective wellness and performance feedback, resulting in impactful systems like PMSys, as described in the paper by Johansen et al. [46]. PMSys undergoes several years of testing and improvements, providing coaches and athletes with valuable statistics to enhance athletic development.

By using feature extraction techniques on this dataset, this thesis aims to provide a more detailed understanding of the performance of the players in the study. Some of the data becomes visualized, allowing for a more intuitive and accessible presentation of the data. The visualization aims to provide insight into the patterns and trends that underlie the players' performance. The data extracted then gets used to create models to forecast injuries.

Furthermore, the insights gained from this research can and will have potential applications in a wide range of areas. Research done in this thesis can get used for further research into areas like injury prevention, player management, game strategy, tactical decision-making, and player recruitment. By identifying key features that have an impact on a player's performance, players, coaches, medical staff, and recruiters can make more informed decisions and can gain a competitive edge.

Overall, this thesis seeks to contribute to the growing field of soccer analytics, by providing insights into a player, or a team's performance and well-being. By extracting features from a dataset containing GPS data and subjective wellness and performance data, visualizing some results, and creating injury prediction models.

1.2 Problem Statement

In the realm of soccer, the utilization of GPS data holds immense potential for injury prediction, performance evaluation, and decision-making enhancement [75]. However, the effective extraction and visualization of relevant features from large datasets, such as the SoccerMon dataset, remains a complex challenge. Furthermore, the prediction of injuries using this data and the determination of the most effective machine learning models for

this task require substantial investigation. Therefore, the overarching problem this thesis seeks to address is:

Can GPS data be used to effectively analyze athlete performance and injuries in professional female soccer?

To address the problem statement, the problem is split into three research objectives.

The raw GPS data is difficult to read and analyze, both because of the vast amount of data, and because of the lack of relevant features available. Therefore, we want to extract features more relevant for answering the problem statement, motivating the first research objective:

RO1: Effectively extract relevant features from raw GPS data, to obtain features more relevant for performance and injury analysis.

In order to aid the analysis of the features extracted in the first objective, we aim to develop a variety of visualization tools that could provide valuable insight into the data, motivating our second objective:

RO2: Create visualizations using features extracted in RO1, subjective data from the SoccerMon dataset and match results.

To delve deeper into the possibilities of analyzing GPS data, our goal is to implement machine learning models for injury prediction, utilizing the data extracted during the first objective, motivating the third objective of this thesis:

RO3: Implement machine learning models to predict injuries based on the features extracted in RO1.

1.3 Scope and Limitations

This thesis aims to extract key features from the raw data obtained from the SoccerMon dataset [59]. The features selected for analysis include, but are not limited to, high-intensity sprints, total distance covered, top speed, average speed, and metabolic power. The data is collected from two different professional female soccer teams in the Norwegian top series over the span of two seasons.

To provide a comprehensive and user-friendly overview of the game and training components, parts of the extracted data is visualized. This visualization will facilitate a more straightforward interpretation and understanding of the performance data. We do not have access to professional athletes or coaches for this study. As a result, obtaining feedback on the visualization tools created is not possible.

Furthermore, this study investigates how the extracted data can be utilized to develop models to predict injuries. The prediction models use one-step ahead forecasting to forecast the injury value for the upcoming day.

However, there are certain limitations to this study. Firstly, the dataset used is only from two teams over two seasons, which means that the findings may not be representative of other teams, or even the same teams in different seasons. Additionally, a lot of the data was collected during the COVID-19 pandemic, meaning players may have been affected both physically and mentally by the disease and lockdowns. This also means that the results may not be representative of other seasons.

All the data being used in this thesis is from the SoccerMon dataset. The dataset has issues with erroneous data points, missing data, and incorrect data. This will have an effect on the results derived in this thesis.

1.4 Research Methods

This thesis primarily employs quantitative and experimental research techniques. Quantitative research methods involve the systematic collection and interpretation of numerical data, allowing for statistical analysis [41]. Experimental research, on the other hand, involves manipulating independent variables to study their effects on dependent variables, which helps establish causal relationships [50].

We analyze the data generated from our experiments using statistical methods to assess the results. Our experimental process and pipeline is developed through a combination of trial-and-error and iterative refinement. We opted for these research approaches since the outcomes of our experiments are uncertain, and we needed to make adjustments based on the findings. Quantitative analysis enables comparison through statistical methods, helping to identify factors like the best data and model setups. Moreover, the use of experimental prototyping guarantees well-defined, testable experiments, while the iterative approach facilitates the step-by-step enhancement of our prototype [50].

1.5 Ethical Considerations

The focus of this thesis revolves around the utilization of wellness, performance, injury, and positional data gathered from professional female soccer teams in Norway [59]. Given the personal and sensitive nature of the data, it is imperative to ensure the protection and privacy of the participants' information. All data has been thoroughly anonymized by eliminating all metadata and employing randomly generated file names, a widely accepted method for data anonymization [93]. Furthermore, each athlete involved in the study has given informed consent, fully

understanding the nature of the data collection and its intended use. This approach ensures respect for the privacy of the participants and guarantees the ethical use of their data.

However, some parts of the analysis in this thesis utilize information that is not publicly available through the SoccerMon dataset. Consequently, some parts of the analysis may not be entirely reproducible using the publicly available SoccerMon dataset. This confidentiality is necessary to preserve personal data about the teams and players involved in the study. Nevertheless, the findings derived from this private data revealed interesting insights, warranting their inclusion in the thesis despite the limitations on reproducibility.

1.6 Main Contributions

In this thesis, our main scientific contribution is answering the main problem statement stated above. This thesis also makes significant strides in developing a versatile feature extraction tool, rendering extracted datasets publicly available, crafting user-friendly visualization tools, and implementing ready-to-use machine learning models with class imbalance handling.

Our key contributions include:

- A flexible feature extraction tool accessible via Jupyter Notebooks.
- User-friendly visualization tools slated for integration into the Soccer Dashboard, an interactive application under development by the PMSys team. These tools will provide invaluable insights for training programs and athlete development.
- Machine learning models for injury prediction that handle class imbalance, complete with preprocessing steps, various model choices, and techniques for addressing class imbalance.

1.7 Thesis Outline

This thesis is organized as follows:

Chapter 1 - Introduction provides an overview of the problems this thesis seeks to address, the scope and limitations, research methods, ethical considerations, and main contributions.

Chapter 2 - Background and Related Work lays out the necessary theoretical knowledge relevant to our thesis. In this chapter there is a look at the history of data gathering in soccer and sports in general. Examples of how data analytics is being used in sports today. Information

about machine learning algorithms, and in depth information about the SoccerMon dataset.

Chapter 3 - Methodology describes the approaches taken in this thesis, the chapter is split into 5 main parts, corresponding to our pipeline. Explaining the approaches taken in data import, data preprocessing, feature extraction, visualization and injury prediction.

Chapter 4 - Implementation takes a deeper look at the implementation of our pipeline. Explaining how we calculate and extract features, how we implement different visualization tools, and how we implement machine learning models for injury forecasting.

Chapter 5 - Results presents the results acquired. The chapter starts by providing some statistical analysis of the new datasets, before each visualization tool is explained. Lastly, a look at the injury forecasting results.

Chapter 6 - Discussion summarizes the thesis, looks at some of the main contributions from the thesis, and propose future work based on our findings.

Chapter 7 - Conclusions The conclusion of the thesis.

Chapter 2

Background and Related Work

2.1 Athlete Health and Performance Monitoring

2.1.1 Wellness Reporting

The process of wellness reporting enables the athlete to evaluate their internal state based on various factors like muscle soreness, stress, fatigue, and sleep, by providing a score for each. These values often combine to generate a total wellness score. According to Wing et al. [94], wellness reporting gains widespread recognition in sports science as a useful metric, surpassing objective measures like heart rate and blood markers. Nonetheless, the efficiency of wellness reporting depends on the specificity of the questions and their ability to provide meaningful data for the given sport [81]. The wellness questions should also be broad enough to be comprehensible for the entire team and not time-consuming. To minimize time, the number of questions should be limited, and the response should preferably be in the form of a numerical score [81].

2.1.2 Training Load

Load monitoring tools can be helpful in identifying how well an athlete adapts to their training program and in providing insight into the risk of fatigue or injury, thereby potentially reducing the likelihood of fatigue or injury occurring. However, fully comprehending and efficiently utilizing training load metrics can be challenging, as only a few of these metrics are supported by substantial scientific evidence and no definitive metric currently exists. It is therefore essential to develop monitoring systems based on what is most relevant to the particular sport and can provide meaningful data about the individual [32].

2.1.3 Injury and Illness

Injuries and illnesses are common occurrences among athletes of varying degrees and can significantly impact their performance and overall well-being. One way to address this issue is to identify patterns and trends that could help prevent undesired outcomes. FIFA implements injury

prevention programs that focus on areas that are prone to injury. Results show that such programs can reduce the overall risk of injury by 34%, and specific exercises like the Nordic Hamstring exercise can lead to a 51% reduction in hamstring-related injuries in the long term, compared to teams without such programs [1]. This highlights the importance of identifying injury patterns and creating preventative programs to significantly reduce the risk of injury.

2.1.4 Positional Data

The use of positional data provides valuable insights into the athletic performance of individuals [72, 78]. However, obtaining precise locomotor movements poses significant challenges, as highlighted by Pettersen et al. [72]. In their research, Pettersen et al. compare GPS technology to LPM (Local Position Measurement), which emits signals to local receivers for triangulation, unlike GPS which only receives signals. While clubs prefer GPS technology for wearable positional tracking, Pettersen et al. suggest that LPM technology might be a more effective alternative. LPM demonstrates greater accuracy, and environmental factors such as cloud cover and the satellite's angle to the ground do not impact the accuracy of data gathered with LPM.

In order to compare the accuracy of GPS and LPM, several studies are conducted. The first study involves six high-level female players who wear two GPS tags and two LPM tags while performing the Copenhagen Soccer Test for Women. The results show that the distance measured by GPS is 11,668 ± 1,072 meters with a coefficient of variation (CV) of 6%, while LPM measures a distance of 10,204 ± 103 m with a CV of 1%. When considering high-intensity runs (HIR), runs with a speed greater than 16 km/h (4,4m/s), GPS measures 612 ± 433 m with a CV of 37.4%, while LPM measures 1238 ± 38 meters with a CV of 3.1%. The study shows that GPS data has a wider spread of data points, especially at higher speeds. The use of two tags of each technology allows for testing of inter- and intra-reliability, and the data shows that the difference in measurement between two tags on the same player using GPS data ranges from 800 to 2,071 meters, while LPM data ranges from 25 to 290 meters. These findings are consistent with Johansen's paper [47], which finds that 19 players obtain an average distance of 10,805 ± 847 meters using GPS data compared to 9,891 ± 974 meters from using LPM data. The second study supports the results found in the first study, with GPS-measured data consistently measuring larger distances for HIR-related tests.

To further test the accuracy of the devices, the players are asked to run along the edge of the soccer field wearing all four tags. The path taken by the athletes is drawn on top of the playing field, resulting in two images that differ significantly. While both images have curved corners, which is expected due to the players not running at an exact 90-degree angle, the LPM image is much more accurate than the GPS image. The GPS struggles

to stay on the edges and keep its lines from curving out, possibly due to various forms of interference such as clouds and fog. Although the tests are conducted at 69.65 degrees north, where the GPS satellite's inclination from orbit is 55 degrees north and south, which means there are no satellites directly overhead, the error of identical tags producing different results occurs for other GPS models at more optimal orbits. The paper concludes that LPM shows superior accuracy over the GPS model, but it remains unclear if the worse accuracy impacts the GPS's usefulness in quantifying the athlete's performance.
The identification of injury patterns and the development of preventive programs show promising results. However, further investigation is required to determine the most effective strategies for reducing the risk of injury in different sports. Machine Learning models are excellent tools to analyze time series data such as injury and positional data. Therefore, they can be instrumental in helping identify injury patterns.

2.2 Machine Learning

The field of machine learning falls under the umbrella term of artificial intelligence (AI) and involves the use of algorithms to detect patterns and learn from data in order to improve accuracy. These algorithms create unique models based on sample data to make predictions or decisions that are not explicitly programmed. The concept of machine learning is around since the mid-20th century and is first coined by Arthur Samuel in the 1950s [80]. Recent advances in hardware and algorithms lead to a resurgence of machine learning over the past decade, with significant improvements in accuracy and scalability. Different types of models are tailored to specific areas and data types, such as convolutional neural networks for computer vision and time series analysis [21], and transformers for natural language processing [81]. In the following sections, we discuss a few different machine learning algorithms, and some terms frequently used in the field.

2.2.1 Supervised Learning

Supervised learning is a prevalent technique in machine learning that relies on labeled data to train algorithms. The labeled data assigns a label or category to each data point, allowing the algorithm to learn and predict outcomes accurately. This method is often used for things like image recognition, natural language processing, and speech recognition [18].

The aim of supervised learning is to approximate a function f that maps an input space X to an output space Y. The algorithm trains by adjusting the model's weights, which involves mapping X to Y. The model is considered a good fit when it accurately predicts the label or category of new, unseen data. Supervised learning is a fundamental concept in machine learning and a powerful tool for various applications. However, it is crucial to

note that the quality of labeled data used for training is critical to the algorithm's performance. Moreover, overfitting can occur if the model is too complex or if there is insufficient data, leading to poor generalization when it comes to new data [95]. Therefore, choosing the right algorithm and data is essential to ensure optimal performance.

2.2.2 Unsupervised Learning

Unsupervised learning algorithms do not have preprogrammed instructions to interpret observations or validate their reasoning. Instead, they rely on the data to find patterns and make sense of the information presented to them [6]. This approach is distinct from supervised learning, where algorithms are given correct outputs and train to recognize patterns in input data that correspond to the correct output.

Clustering is one of the most popular techniques in unsupervised learning [29]. It groups similar data points together based on their features. Because the algorithm lacks prior knowledge of the data, it tries to form clusters based on the similarity of the data points. To summarize, unsupervised learning is a powerful technique that identifies patterns and structures in data without the need for preprogrammed instructions.

2.2.3 Overfitting and Underfitting

Overfitting occurs when a statistical model fits too well to the training data. This leads to poor performance on other unseen datasets, as the model's ability to generalize diminishes [43]. For example, an overfitted model produces exact lines separating two data classes, as shown in the rightmost image in Figure 2.1.

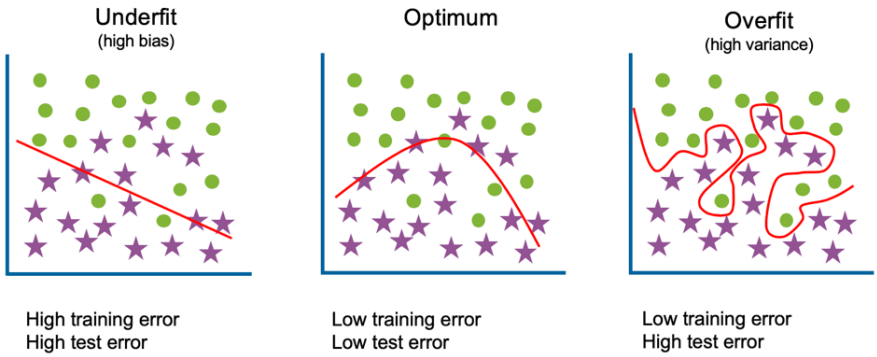


Figure 2.1: *Examples of Overfitting, Underfitting and Optimum. [66]*

Figure 2.1 shows that when the model becomes too complex, it starts fitting the noise in the data rather than the underlying patterns, creating overfitting. The figure also shows how underfitting can be a problem. Underfitting occurs when a model is too simple and cannot capture the underlying patterns in the data. This usually happens because there is

not enough data to train the model properly, or because the model is not complex enough to handle the complexity of the task.

To avoid overfitting and underfitting, it's important to carefully select the appropriate model architecture and hyperparameters. Additionally, using techniques such as regularization on the data helps. Lastly, and most importantly, it is crucial to have a sufficient amount of diverse and representative data to train the model.

2.2.4 Logistic Regression

Logistic regression is an algorithm that establishes a relationship between an independent variable and a binary dependent variable to predict the probability of certain categorical outcomes[39]. It is a statistical method employed in data science and machine learning for predictive analysis.

The independent variable, also known as the predictor or explanatory variable, remains unchanged by other variable fluctuations. In contrast, the dependent variable, which is binary, alters with changes in the independent variable. The regression model predicts the probability of the dependent variable being one of the two categories.

The multiple logistic regression equation is an extension of the simple logistic equation, $p = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$, with the addition of weights and inputs for the various features represented by p(n)x(n). The multiple logistic regression formula looks like this:

$$p = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p)}}$$

The machine learning model employs this formula and various weight values to fit curves to the data. In order to identify the curve that best fits the data, the model assesses different weight combinations that maximize the relationship between the variables.

2.2.5 Decision Trees

Decision trees find numerous applications in machine learning, covering both classification and regression tasks [45]. In decision analysis, decision trees serve as a visual and explicit representation of decisions and decision-making processes [30].

Real-world datasets typically contain numerous features, resulting in more complex trees. However, the simplicity of this algorithm is evident, with clear feature importance and easily discernible relationships [30]. Regression trees function similarly but predict continuous values, such as house prices. Generally, decision tree algorithms are known as CART, or Classification and Regression Trees.

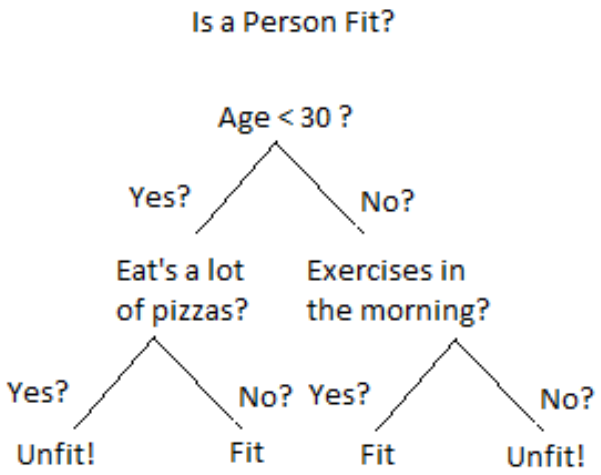


Figure 2.2: *Example of a decision tree. [20]*

Underlying processes in decision tree growth involve selecting features and splitting conditions, as well as determining when to stop. Trees tend to grow arbitrarily, necessitating trimming to maintain simplicity. A common splitting technique is recursive binary splitting. This method considers all features and evaluates various split points using a cost function [30]. The split yielding the lowest cost is chosen.

For a dataset containing 3 data points, the initial split or root considers all features, dividing the training data into groups accordingly. With three features, there are three potential splits. A function calculates the accuracy cost of each split, and the split with the lowest cost is selected [30]. The recursive nature of this algorithm allows for further subdivision of groups using the same strategy. This approach, known as the greedy algorithm, focuses on minimizing cost, making the root node the best predictor or classifier.

Split Cost Evaluation

Cost functions for classification and regression aim to identify the most homogeneous branches or groups with similar responses [30]. This makes sense as it ensures that a test data input follows a specific path.

Regression

$$\sum(y - \text{prediction})^2$$

The decision tree starts by splitting each feature in the training data. The mean response of specific training data inputs within a group is considered the prediction for that group. The function above calculates the cost for

all splits, with the lowest-cost split being chosen. Another cost function involves reducing the standard deviation [30].

Classification

$$G = \sum p_k(1 - p_k)$$

A Gini score indicates the quality of a split based on the homogeneity of response classes in the groups created. Here, pk represents the proportion of identical class inputs within a group. Perfect class purity occurs when a group contains only inputs from the same class, resulting in a Gini score of 0. Conversely, a node with an even split of classes in a group has the worst purity [30].

2.2.6 xGBoost

Extreme Gradient Boosting, or xGBoost for short, is an optimized implementation of the popular gradient boosting algorithm [13, 14]. This powerful machine learning approach becomes widely used for supervised learning tasks like regression, classification, and ranking problems. Created by Tianqi Chen with exceptional performance and scalability in mind, xGBoost sees widespread adoption in various machine learning competitions as well as real-world applications.

xGBoost utilizes the principle of boosting, an ensemble learning method that pools multiple weak learners (usually decision trees) to create a powerful learner. The algorithm trains multiple base models sequentially, each new one correcting errors made by its predecessors [13, 14].

Gradient boosting in xGBoost involves iteratively adding new decision trees while updating their weights using gradient descent [13, 14]. The goal is to find the optimal combination of weak learners that minimizes the loss function, producing a robust and accurate model.

In conclusion, xGBoost is an impressive and efficient machine learning algorithm that demonstrates its worth in numerous applications and competitions. Its speed, performance, and versatility make it a valuable resource for data scientists and machine learning practitioners working on regression, classification, and ranking problems.

2.2.7 Neural Networks

Neural networks are a subset of machine learning (ML) that employs an artificial approach to mimic the way biological neurons, such as those found in the human brain, communicate with each other. These networks consist of an input layer, an output layer, and a variable number (N) of hidden layers in between. Each artificial neuron in the network can be conceptualized as a node connected to other nodes [31].

The first trainable artificial neural network, known as the Multilayer Perceptron (MLP), is introduced by Rosenblatt [77] in the 1950s. The MLP features only a single layer between the input and output layers, utilizes a threshold activation function, and is designed to serve as a linear classifier. This early neural network design significantly differs from modern neural networks, which employ non-linear activation functions [69]. The use of non-linear activation functions allows contemporary neural networks to learn from a broader range of data and generate more complex decision boundaries.

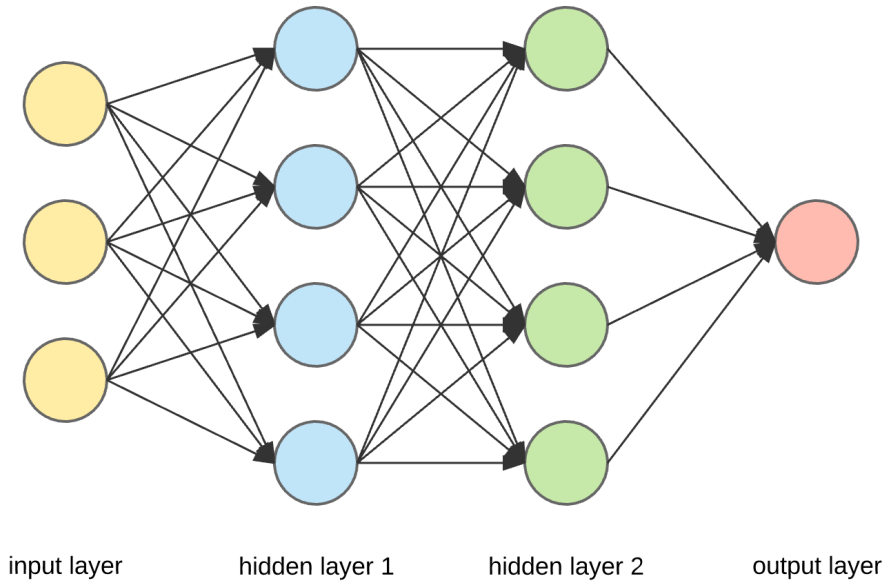


Figure 2.3: *Example of a Neural network. [63]*

The learning process of a neural network primarily consists of three stages: forward propagation, error computation, and backpropagation. These stages work together to iteratively enhance the network's overall performance [4].

Forward Propagation

During forward propagation, input data travels through all the layers of the network, ultimately generating a prediction as output. This process begins with each node receiving outputs from all the nodes in the previous layer, which are then combined with the node's unique set of weights. Subsequently, an activation function is applied to compute the hidden activation for each node, which serves as input for the following layer of nodes [36].

Error Computation

Once forward propagation yields a prediction, it is essential to compare this output with the actual prediction that the model is intended to produce. This comparison, known as error calculation, is achieved using a loss function, such as mean squared error $\sum_{i=1}^{D}(x_i - y_i)^2$ [91]. The loss function is employed to adjust the network's weights, enhancing the accuracy of the mapping from X to Y using backpropagation.

2.2.8 Recurrent Neural Networks (RNN)

A Recurrent Neural Network (RNN) is a type of artificial neural network designed to handle sequential or time series data [83]. These advanced learning algorithms are widely used in ordinal or temporal problems such as language translation, natural language processing (NLP), speech recognition, and image captioning. They integrate into popular applications like Siri, voice search, and Google Translate [56]. Like feedforward and Convolutional Neural Networks (CNNs), RNNs learn from training data, but they are unique due to their "memory," which allows previous inputs to influence the current input and output. Unlike traditional deep neural networks that consider inputs and outputs as independent entities, RNN outputs depend on earlier elements in the sequence. However, unidirectional RNNs cannot incorporate future events into their predictions [83].

One unique feature of recurrent networks is their shared parameters across each layer. While feedforward networks have distinct weights for each node, RNNs share the same weight parameter within each layer [83]. Nonetheless, these weights still adjust through backpropagation and gradient descent for reinforcement learning.

RNNs employ the Backpropagation Through Time (BPTT) algorithm to calculate gradients, which is tailored for sequence data and slightly different from conventional backpropagation. BPTT's core principles are similar to traditional backpropagation, as the model trains by computing errors from the output layer to the input layer. These calculations enable appropriate parameter adjustment and fitting. BPTT's distinction lies in summing errors at each time step, unlike feedforward networks that do not require error summation since they don't share parameters across layers [83].

RNNs often face two challenges: exploding gradients and vanishing gradients. These issues stem from the gradient size, which represents the slope of the loss function along the error curve. When the gradient is too small, it continually shrinks, updating weight parameters until they become negligible—or zero—meaning the algorithm stops learning. Exploding gradients occur when the gradient is too large, resulting in an unstable model. In this case, model weights grow excessively, eventually being represented as NaN. One way to address these problems

is by reducing the number of hidden layers in the neural network, thus simplifying the RNN model [83].

2.2.9 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) designed to address the shortcomings of traditional RNNs, especially the vanishing gradient problem [37, 90]. This issue arises when training deep RNNs, leading them to have difficulty learning long-range dependencies in data, especially sequences. LSTMs find widespread application in natural language processing, speech recognition and time series prediction due to their capacity for handling such sequences with long-term dependencies.

LSTMs utilize an innovative cell structure that helps maintain data for extended periods. At its core, an LSTM cell consists of a memory cell that stores information for long periods. An LSTM cell also comprises three essential gates: input gate, forget gate, and output gate. These controls regulate information flow into and out of the cell, enabling the network to learn when to preserve or discard information as required [90].

Input Gate: The input gate controls how much of a new input is added to the cell state. It uses a sigmoid activation function to produce a value between 0 and 1, representing the proportion of the new input that will be incorporated. Simultaneously, a separate path applies the hyperbolic tangent (tanh) activation function to the input to scale it between -1 and 1 [90].

Forget Gate: The forget gate determines which information from a cell state is retained or discarded. It uses a sigmoid activation function to generate a value between 0 and 1. A value closer to zero indicates that the information will be forgotten, while one closer to 1 indicates retention of that same data [90].

Output Gate: The output gate controls the flow of information from a cell state to its output. It uses a sigmoid activation function to generate a value between 0 and 1, determining how much information from that cell should be output. Furthermore, it applies a tanh activation function to scale values between -1 and 1 [90].

LSTM cells are designed with the capacity to learn when to store, update or output information from their current state. This trait allows the network to accurately capture long-range dependencies in data. As a result, LSTMs are ideal for many sequence-based tasks and applications [90].

2.2.10 Gated Recurrent Units (GRUs)

Gated Recurrent Unit networks are an alternative type of Recurrent Neural Network architecture introduced as a simpler option to Long Short Term Memory networks [16]. Like LSTMs, GRUs address the vanishing gradient problem found in traditional RNNs, enabling them to learn long-range dependencies among sequential data. Due to their capability for handling sequences effectively and with fewer parameters compared to LSTMs, GRUs become popular in various applications such as natural language processing, speech recognition, and time series prediction due to their capability for dealing with sequences more efficiently and having fewer parameters compared to their LSTM counterparts [90].

GRU networks simplify LSTM architecture by only having two gates compared to LSTM's 3. GRUs two gates are the update gate and reset gate. By eliminating the output gate as well as the cell state from GRU's computation requirements, it becomes faster to train and more efficient in certain circumstances [16].

Update Gate: The update gate utilizes the functionality of both the input and forget gates from an LSTM model, determining how much of the previous hidden state should be retained and which part should be used for new candidate state generation. It uses a sigmoid activation function to produce values between 0 and 1, with 0 signifying complete replacement of the previous hidden state and 1 signifying retention [16].

Reset Gate: The reset gate controls how much of a past hidden state should be considered when creating the new candidate state. Like the update gate, it utilizes a sigmoid activation function to generate values between 0 and 1. A value closer to zero implies minimal influence from the past hidden state on the new candidate state, while one closer to 1 indicates strong influence from it [16].

GRUs simplify LSTM architecture by condensing input and forget gates into one update gate and discarding the output gate. As a result, GRUs have fewer parameters, making them computationally more efficient and quicker to train than LSTMs; however, their expressiveness may not match up as well, potentially leading to slightly reduced performance on certain complex tasks. Overall, GRUs make sense as an option for sequence-based tasks where computational efficiency and training time are essential considerations [16, 90].

2.2.11 ROCKET

ROCKET (Random Convolutional Kernel Transform) is not a traditional Convolutional Neural Network (CNN), yet it borrows some concepts from CNNs. ROCKET is an algorithm for time series classification that is both computationally efficient and highly accurate, proposed by Angus

Dempster et al. [21]. ROCKET generates random convolutional kernels that are applied to time series data to form feature maps. Global max pooling and proportion of positive values (PPV) are then applied to these feature maps in order to extract features, which then go on to form input for linear classifiers such as Ridge regression or logistic regression [21].

ROCKET utilizes convolutional operations, but does not follow the traditional CNN architecture and training process. Instead of learning convolution kernels through backpropagation, ROCKET randomly generates them, making the algorithm faster and more computationally efficient [21]. In conclusion, ROCKET is not a traditional CNN, but it borrows concepts from CNNs such as convolution and pooling to efficiently classify time series data [21].

2.3 SoccerMon Dataset

The use of scientific methods in sports increases over the past three decades, aided by the availability of athlete quantification data, sensor technology, and advanced analytic software. In soccer, where performance is affected by multiple factors, there grows a focus on mental and physical parameters that can impact training response and injury prevention. However, data availability remains a significant challenge, and there is a need for more research in women's soccer. To address this, the creators of the SoccerMon dataset propose a dataset containing both subjective and objective data from professional teams in the Norwegian female elite soccer league, collected over the 2020 and 2021 seasons using wearable tracking equipment and athlete monitoring systems [59]. The dataset includes information on e.g. training load, game performance, wellness, sickness, injuries, and objective measurements of locomotor activity during training and matches. The creators of the dataset hope that this dataset enables more research on the effects of various mental and physical parameters of female soccer and contributes to individualized training, better player selection, and injury prevention.

2.3.1 Collection Methods

In SoccerMon, the data collects from two teams in the female elite soccer league in Norway over a period of two years (2020 and 2021) using various methods to obtain both subjective and objective data. The PMSys system [46] creates to facilitate efficient and rapid systematic longitudinal tracking of athletes' phenotypic and self-reported wellness, performance and training load parameters. This does to support the logging of subjectively perceived parameters for athletes. The PMSys app is available for both Android and ios systems and allows athletes to easily interact with the system. It is a smartphone-based tool that enables players to report important parameters such as session Rating of Perceived Exertion (sRPE), wellness, injury, illness, session participation, game evaluations, and in

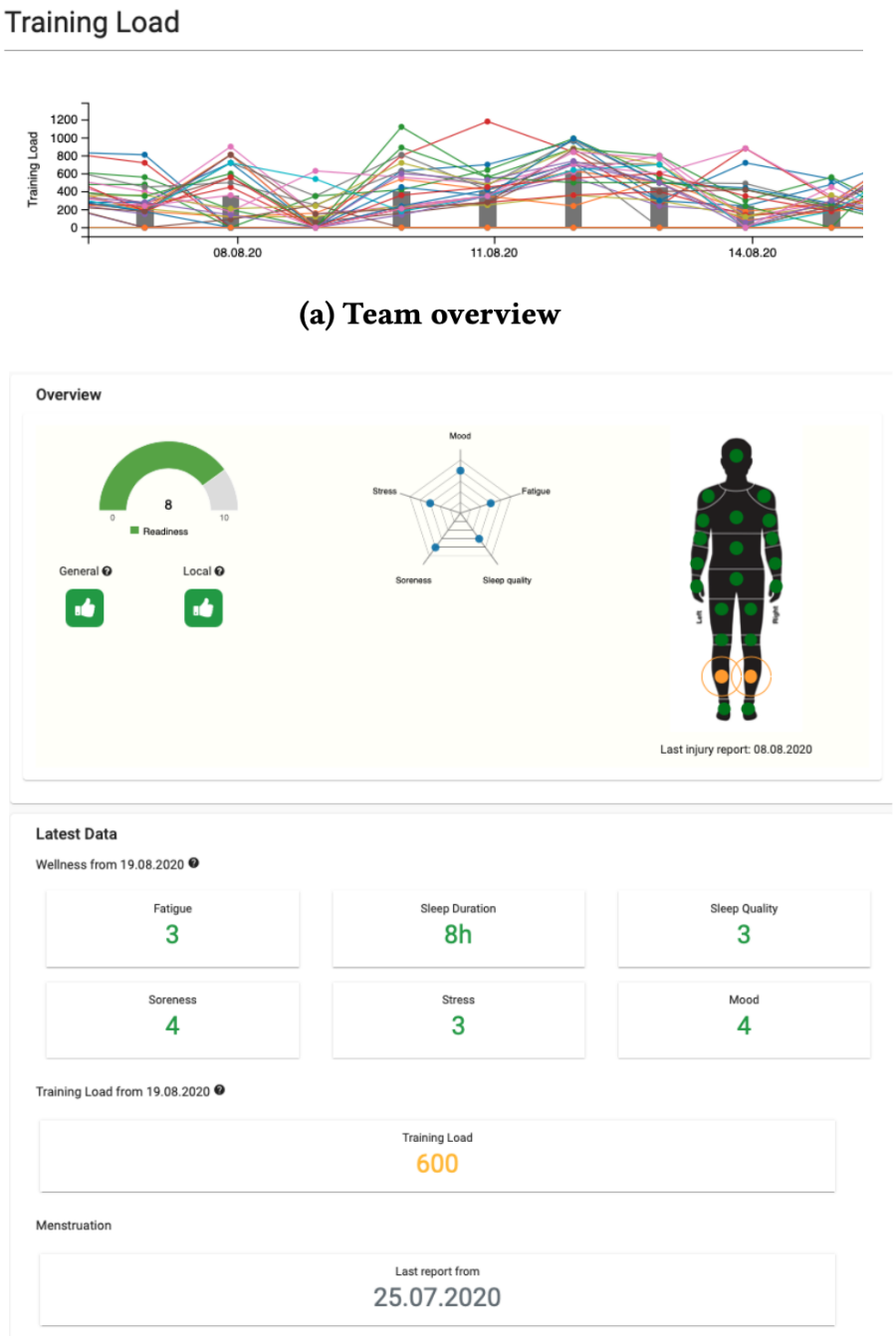
light of the COVID-19 pandemic, infection symptom checks. The system operates on the principle that subjective reports should be captured in real-time with minimal effort, while they are fresh and relevant. For example, the athlete can report wellness metrics within the narrow time window after they wake up and before the first morning training session begins.

The Trainer Portal provides a web-hosted Single-Page Application (SPA) that enables team personnel, such as coaches and physicians, to access and interact with athlete data. The portal displays submitted data from team members who have granted access, both in a team view and a single-player view. The team view includes visualizations of injuries, illnesses, and session participation (see Figure 2.4a), along with important training load indicators like daily and weekly load, acute load, chronic load, acute chronic workload ratio, monotony, and strain. The single player view offers individual details (see Figure 2.4b). The portal also includes a feature for sending push messages, either directly or on a schedule, as reminders to report.

2.3.2 Contents

Subjective Metrics

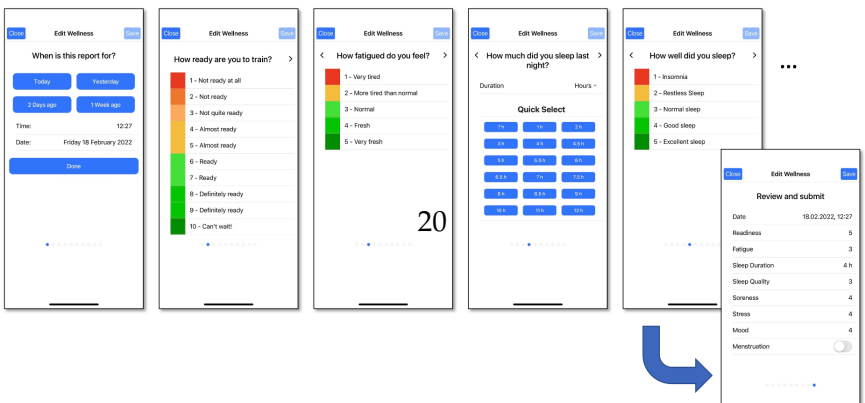
The initial section of the SoccerMon dataset comprises subjective reports of wellness, performance, and training load, recorded by athletes via the PMSys system. The PMSys app features a questionnaire, illustrated in Figure 2.5, containing parameters for wellness such as fatigue, readiness to train, sleep quality, soreness, stress, mood, and optionally, menstruation. Readiness rates on a scale of 1 to 10, sleep length reports in hours, and menstruation indicates with a single tick button, while the remaining parameters use a balanced Likert-scale from 1 to 5. The scales accompany textual explanations to ensure a consistent interpretation of the numerical ratings. Athletes are to complete the wellness questionnaire once per day. Figure 2.6 depicts a comparable questionnaire for recording training load, which is to be reported after every training session or activity, including games. The questionnaire captures the length and type of the session and a rating of perceived exertion (1 to 10) for session difficulty.



(a) Team overview

(b) Single player

Figure 2.4: *An illustration of how the Trainer Portal looks in the PMsys application (courtesy of ForzaSys).*



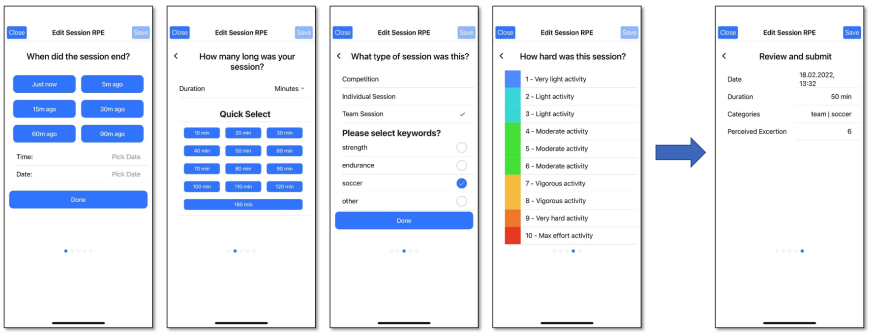


Figure 2.6: *An illustration of how a player can report sRPE in the PMsys application. (courtesy of ForzaSys)*

This data allows for the calculation of several sports scientific measures, including sRPE, daily load, weekly load, acute training load, chronic training load, acute chronic workload ratio, monotony, and strain. In addition to wellness and training load, injury and illness data also collect. These reports record pain location and severity and symptoms, respectively, with medical personnel responsible for drawing conclusions. Injury reports submit using a body silhouette (refer to Figure 2.7), where a single tap indicates minor pain (orange), and a double tap indicates major pain (red). The data collection process is similar to the aforementioned examples.

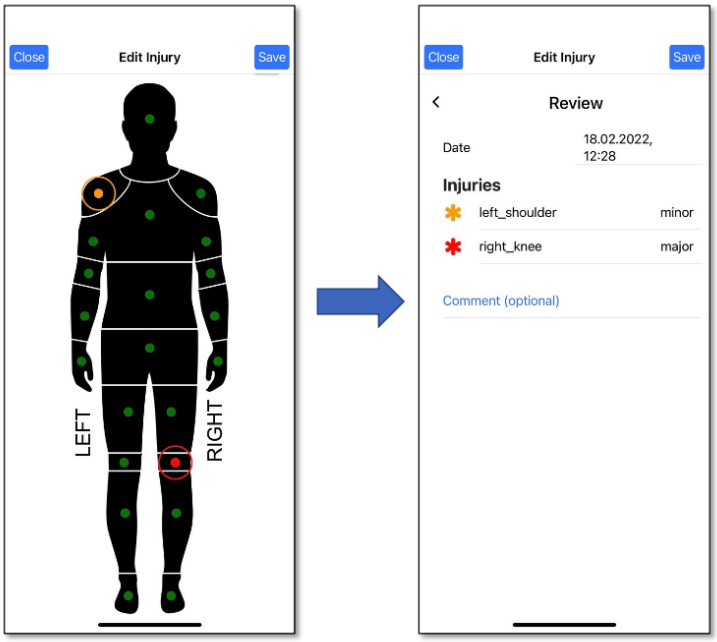


Figure 2.7: *An illustration of how a player can report injuries in the PMsys application (courtesy of ForzaSys).*

Objective Metrics

In order to measure movement patterns and physical demands in elite women's soccer, both teams are provided with the STATSports APEX system, which is approved by FIFA [84]. The system, which is built into a vest worn by the players (as shown in Figure 2.8), includes 10 Hz multi-GNSS augmented units that can track multiple satellite systems simultaneously (such as GPS, GLONASS, Galileo, and BeiDou). Additionally, the system is equipped with a tri-axial accelerometer, and tri-axial gyroscope.

Each player wears a GPS unit, which is the size of a matchbox and is tightly fitted on their upper back, during both training and matches as illustrated in Figure 2.8. In addition to the GPS unit, players also have the option to wear a chest strap to monitor their heart rate. To ensure consistency, each player uses the same GPS unit throughout the season, which helps minimize inter-device errors. After each session, the data recorded by the units are retrieved and uploaded to the respective club's laptop using the STATSports Sonra 2.1.4 software. For the data to be used for analysis purposes, the raw data (in csv-format) is downloaded and saved in a local folder, where files for a particular year and month (e.g. 2020-06) are zipped into their respective subfolder. Each zipped folder containing data for a given year-month is then uploaded to OSF.

Figure 2.8: *STATSports GPS tracker in vest during training. (courtesy of ForzaSys)*

2.4 Previous Work on Time Series Forecasting

Forecasting future values in time series sequences is an essential task across various domains, including finance, meteorology, medicine, and sports science. Time series analysis aims to extract meaningful insights from

historical data and forecast future data points. In this section, we review some of the current work conducted in this area, focusing on traditional statistical methods, machine learning, and deep learning approaches.

2.4.1 Machine Learning Approaches

Machine learning techniques are increasingly applied to time series prediction in recent years. These methods can automatically learn complex patterns from the data without relying on strict statistical assumptions. Some commonly used machine learning algorithms for time series prediction include support vector machines (SVM) [35], random forests [10], and gradient boosting machines (GBM) [25]. Feature engineering plays a crucial role in these models, often requiring domain-specific knowledge to transform the raw time series data into meaningful input features.

2.4.2 Deep Learning Approaches

Deep learning methods, especially recurrent neural networks (RNN) and their variants, such as long short-term memory (LSTM) [37] and gated recurrent units (GRU) [16], have demonstrated remarkable performance in time series prediction. These models can effectively learn temporal dependencies and capture complex patterns within the data.

2.4.3 Multivariate and Multi-step Forecasting

In many real-world applications, it is necessary to consider multiple input variables or predict multiple time steps ahead. Multivariate time series prediction involves incorporating multiple related time series as input features, while multi-step prediction aims to predict multiple future values simultaneously. Several techniques are proposed to handle these challenges, including multi-output regression [86], direct and iterative strategies [8], and the use of encoder-decoder architectures [85].

2.4.4 Readiness Forecasting using the SoccerMon Dataset

In 2022, Ragab et al. did a study where they employed machine learning to predict readiness values, using the SoccerMon dataset [74]. They employed a LSTM model to predict the self reported readiness value for players. They concluded that **1:** When training on the entire team and predicting for a single player, the overall accuracy was higher for Team A, which exhibited greater player consistency, while it was lower for Team B, where player consistency was lacking. **2:** The prediction inaccuracy increased as the prediction period extended into the future. As a result, an output window of 1 tracked the peaks more accurately than an output window of 7. Expanding the input window did not noticeably improve the accuracy of predicted values. **3** It was noted that having more data did not necessarily enhance prediction accuracy for models trained on either the whole team or individual players. Nonetheless, training on the entire team using one

year of data yielded better results than training on a single player with two years of data. This suggests that a shorter duration of collected data for a team may be more advantageous than a longer time frame. **4:** Altering the hyperparameters did not lead to substantial differences. The only significant factor was shuffling, which slightly improved the results when activated.

2.5 Predicting Injuries using GPS Data

GPS tracking offers the ability to monitor an athlete's speed and distance during games and training sessions. This information converts into training load metrics, which then get employed to create athlete profiles for future performance and injury prevention. The study by Rossi et al. [78] suggests approaches for effectively using GPS data to analyze professional soccer players by extracting 12 features for a multi-dimensional model that forecasts potential injuries based on recent training load.

The 12 workload GPS features divide into three categories:

- Kinematic features that describe an athlete's overall movement
- Metabolic distance features that quantify an athlete's energy expenditure
- Musculoskeletal load features that provide a general measure of the strain on the body. Alongside these 12 features, another 43 features are included, such as personal data (age and injury history), Exponential Weighted Moving Average features, Acute Chronic Workload Ratio features, MSWR features indicating the monotony of workload features, and the connection between a current training session and previous injury.

These features constitute the training dataset and undergo further processing to identify the most relevant features by reducing the feature space dimensionality, minimizing the risk of overfitting. This is achieved through a feature selection process called Recursive Feature Elimination with Cross-Validation (RFECV) [60], which results in a subset of features with the highest score on the validation set.

Various models are tested using the dataset, with the decision tree model performing the best, achieving a recall and precision of 0.8 and 0.5, respectively. This indicates that the model identifies injuries 80% of the time and accurately classifies games/training sessions as injuries 50% of the time.

Besides being tested with data from an entire season, the model also gets tested on another season, receiving new data for every training session/game. This is conducted to evaluate the model's performance with

limited data. The paper reports poor performance during the first few weeks, but from week six until the season's end, the model identifies nine out of 14 injuries with a precision of 0.56. This suggests that the model becomes efficient after just a couple of months of data collection.

2.6 Chapter Summary

This chapter provides a comprehensive overview of machine learning, a subset of artificial intelligence (AI), and its applications in the context of women's soccer. It discusses the historical background of machine learning and its resurgence due to advancements in hardware and algorithms. The chapter introduces supervised and unsupervised learning techniques, as well as the concepts of overfitting and underfitting, which can impact a model's performance. To avoid overfitting and underfitting, the chapter advises selecting appropriate model architecture, hyperparameters, and data regularization. It also introduces various machine learning algorithms, including logistic regression, decision trees, xGBoost, neural networks, Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), and ROCKET.

In this research, all the mentioned algorithms are selected to address injury prediction in women's soccer. The rationale behind choosing these algorithms is as follows:

- **Logistic regression:** A simple yet effective technique for predicting binary outcomes, which can be useful for analyzing relationships between player statistics and team performance.
- **Decision trees:** A powerful method for handling categorical data and creating interpretable models, allowing insights into the factors affecting soccer outcomes.
- **xGBoost:** A robust, ensemble-based algorithm that can handle large datasets and provide accurate predictions, making it suitable for analyzing complex patterns in women's soccer data.
- **LSTM and GRU:** These algorithms are particularly effective for handling time-series data and can capture the temporal dynamics of soccer matches, enabling prediction of match outcomes and player performance over time.
- **ROCKET:** A recent addition to the machine learning toolkit, this algorithm is designed for efficient and accurate time-series classification, making it an attractive choice for analyzing women's soccer data.

By using this diverse set of algorithms, there is a comprehensive address of the challenges of predicting injuries in women's soccer.

The chapter also focuses on athlete health and performance monitoring in women's soccer, covering topics such as wellness reporting, training load, injury and illness prevention, positional data, and the SoccerMon dataset. Wellness reporting allows athletes to evaluate their internal state, while load monitoring tools help assess adaptation to training programs and injury risk. Positional data, gathered via GPS, provides insights into athletic performance. The SoccerMon dataset, containing both subjective and objective data from professional teams in the Norwegian female elite soccer league, aims to enable more research on female soccer performance. The chapter concludes by exploring the use of GPS data to predict injuries, with the decision tree model proving to be the most effective in Rossi et al [78]. [78] study.

In conclusion, there is a growing need for better analysis of athlete performance in women's soccer to ensure optimal training, injury prevention, and overall player well-being. Machine learning techniques show great promise in addressing these needs by providing deeper insights into the complex interactions between various performance factors. In the next chapter, there is a description of the approach to leveraging machine learning algorithms for enhancing athlete performance analysis and monitoring in the context of women's soccer.

Chapter 3

Methodology

The significance of data visualization tools and athlete tracking explains in earlier chapters. Additionally, various machine learning models, data processing techniques, and prior research related to athlete tracking, performance, and injury examine.

In this chapter, we look at the proposed pipeline for the data. How to extract features from the raw dataset, the decision-making process for selecting visualizations, and a pipeline that incorporates GPS data and machine learning for forecasting injuries in soccer players.

3.1 Proposed Pipeline

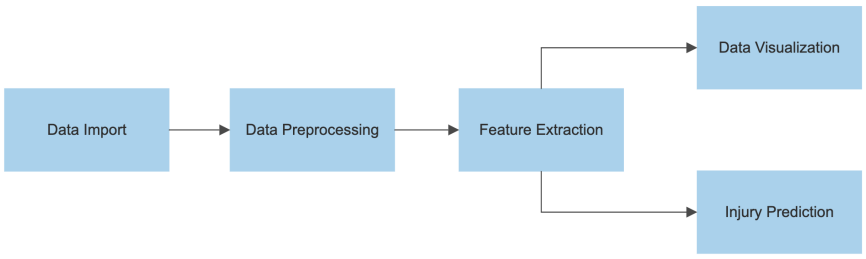


Figure 3.1: *Proposed pipeline.*

Our proposed pipeline comprises the following modules/steps:

- **Data Import:** Downloads, and iterates through all the raw data files. Import parts of the raw data to use in the coming steps in the pipeline.
- **Data Preprocessing:** Cleans and sanity checks the imported data.
- **Feature Extraction:** Extracts features from the preprocessed data for further use in visualization and injury forecasting.
- **Data Visualization:** Creates visualization based on features extracted in the previous step, raw GPS data, subjective performance data, and match results.

- **Injury Prediction:** Implements methods to predict injuries, utilizing the features extracted in step 3, and injury data.

3.2 Data Import

In this section, we discuss the tools, data structure, and importing process used in the data import step of our pipeline. Firstly, the PyArrow library is explained, then we look at the data structure of the raw data. Lastly, we look at how the data is imported and stored before preprocessing.

3.2.1 Tools

All code for this thesis is developed in Python and executed through Jupyter Notebook, using several Python libraries for additional features. Here and in subsequent sections regarding tools used in visualization and injury prediction, the most important tools are explained.

PyArrow

PyArrow is a Python library developed to provide an efficient memory interface for Arrow's in-memory columnar data format, targeting modern big data and data science applications. Additionally, its data interchange capabilities facilitate seamless interaction across programming languages and systems [5].

PyArrow, as part of the Apache Arrow project, provides key features like memory efficiency, interoperability with other languages and systems, high performance I/O operations, integration with Pandas, and support for popular file formats. These features make PyArrow an indispensable solution for data processing tasks requiring rapid computations.

3.2.2 Data Structure

The SoccerMon dataset is stored in multiple different folders on the OSF platform. OSF is a free, open-source web application that connects and supports the research workflow, aiming to enable scientists to increase the efficiency and effectiveness of their research [64]. There are two main folders for the data: Objective and Subjective. The Objective folder contains two subfolders, TeamA and TeamB. Each of these folders contains the subfolders 2020 and 2021. Under each of these folders is a folder for each month of the year, and under each month is a zip file for each date. The zip files contain one parquet file for each player for that specific day.

The subjective data is in the Subjective folder. There are two main folders here: the per-feature folder and the per-player folder. Under the per-feature folder, there are multiple CSV and JSON files containing all the data for all players on both teams for both years. There are files like injuries.csv and duration.json, which contain data the players have registered in the

PMSys application. Under the per-player folder, there are two folders: one for Team A and one for Team B. Under each of these are five folders: Daily-Features, Game-Performance, Illness, Injuries, and Session-Features. Under each of these folders, there is one CSV file for each player containing all the subjective data corresponding to the category of the parent folder. A compressed version of the file structure is shown in Figure 3.2

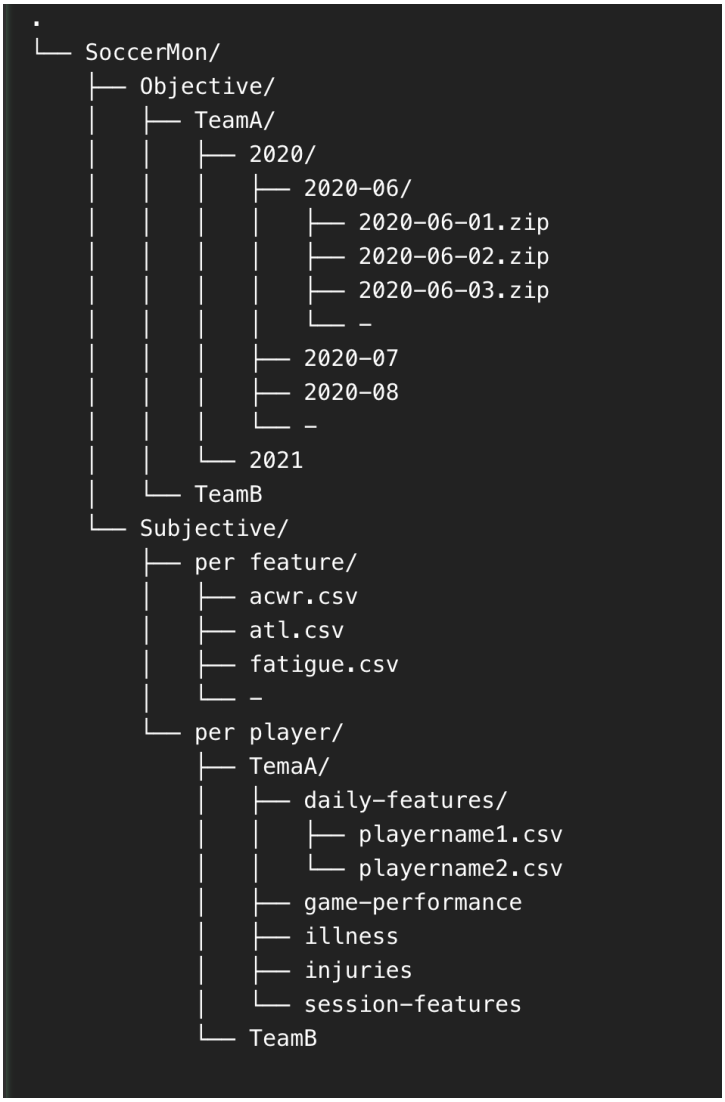


Figure 3.2: *Compressed visualization of the SoccerMon dataset structure*

3.2.3 Importing Data

To extract features from the dataset, it is essential to import the raw data, spread across multiple files, into Python. The feature extraction tool specifically targets GPS data, and thus, only the GPS data is imported at this stage.

The process starts by generating a comprehensive list of all the file paths. This is accomplished using nested loops, which produce a list that includes every potential file path. Following this, the pathlib module is employed to iterate through all accessible files.

Throughout this procedure, data from each file is selectively imported and saved as a dataframe for subsequent use. As outlined in chapter 2, the objective data includes information from a GPS tracker, accelerometer, and gyroscope. Since the gyroscope records values at 100Hz and the GPS tracker and accelerometer record values at 10Hz, there are multiple duplicate entries for the GPS tracker and accelerometer. To avoid importing duplicate values, only every 10th datapoint is imported.

3.3 Data Preprocessing

After importing the raw data into python, several preprocessing steps are taken to clean and sanity check the data before feature extraction. A more in depth look at the preprocessing steps taken, why they are implemented, and how filtering parameters are chosen is explained in chapter 4.

The quality of GPS data is crucial for accurate analysis and reliable insights in any data-driven application. Raw GPS data, however, often contains errors and inconsistencies that can compromise the integrity of the results. Therefore, it is vital to identify and remove bad data from the raw GPS datasets before conducting further analysis. This section discusses the importance of this process.

First, inaccurate GPS data can lead to incorrect conclusions and misguided decision-making. In sports applications, for example, erroneous data may result in incorrect assessments of player performance or fitness levels, which can directly impact coaching decisions and team strategy.

Second, the presence of erroneous data can impact the performance of machine learning models. Many models rely on the quality of input data to make accurate predictions or classifications. Feeding models with erroneous or inconsistent data can decrease their effectiveness and result in suboptimal performance. By cleaning the raw GPS data, we can improve the performance and reliability of our models.

Last, removing erroneous data can streamline the data analysis process by reducing noise and complexity in the dataset. Clean data simplifies the identification of trends and patterns, leading to more efficient data processing and a better understanding of the underlying phenomena. Moreover, it can help save computational resources and reduce processing time by eliminating unnecessary or redundant data points.

Based on these arguments, we decide to use metadata from the raw data to create filters to remove erroneous data before feature extraction. The raw GPS dataset contains metadata about the quality of the collected data, such as quality of the GPS signal, horizontal accuracy and horizontal dilution of precision. By implementing thresholds to this data, erroneous data is removed.

3.4 Feature Extraction

In order to develop visualization tools and datasets for machine learning models, we create two new datasets featuring processed and more relevant attributes compared to the raw data. Ultimately, we create two distinct datasets. The Session dataset is designed to provide an overview of each player's performance during a single session. This dataset is used in many of the visualization tools, as it is great for seeing team averages, player development over time, and seasonal changes in team and player performance. It also serves as the base for the dataset used in the machine learning models.

The second dataset we create is the High-Intensity Run (HIR) dataset. This aims to offer a deeper understanding of when and how players execute high-intensity runs. This dataset enables the generation of intriguing visualizations. As discussed in chapter 2, HIRs have been associated with injuries [57]. Therefore, examining the quantity of HIRs and their related data potentially offers valuable insights into player performance and injury risks for players, coaches, and medical staff.

3.4.1 Tools

Pandas

Pandas is an open-source data manipulation and analysis library, developed for the Python programming language by Wes McKinney in 2008 [58, 67]. First released for public usage in 2008, Pandas provides data structures and functions necessary for seamless analysis of structured data structures.

Pandas allows users to easily manage missing data, apply custom transformations to datasets, generate informative plots and graphs, as well as make statistical analyses with ease. With built-in statistical functions that seamlessly integrate into popular libraries, Pandas is an indispensable asset when working with data in Python.

3.4.2 Session Dataset

The Session dataset is created by us using feature extraction to obtain a comprehensive understanding of each player's performance and physical activity during individual training sessions or matches. Organized on

a per-player-per-session basis, this dataset contains information such as player name, session duration, total distance run, and more. The dataset is created because the raw data does not contain easily understood data, and because the user can get a better overview of a session, than when using the raw data. The dataset has multiple potential use cases:

Analyzing session data helps coaches and trainers design customized training programs tailored to each player's specific needs, strengths, and weaknesses. This leads to more effective skill development, better fitness levels, and overall improved performance on the field [70]. The dataset allows coaching staff to track the workload of each player over time, which helps in managing fatigue and preventing over training. By understanding each player's workload, coaches make informed decisions regarding player rotation, rest days, and recovery strategies.

The dataset is also used as a base for multiple different visualization tools, which help players, coaches, and analysts gain insights into individual and team performance, and identify patterns or trends that might be useful for strategic decision-making. The session dataset is employed for injury prediction as well. By analyzing various performance metrics and understanding each player's workload, injury risks are assessed and proactive measures are taken to minimize the likelihood of injury occurrence.

In conclusion, the Session dataset offers valuable insights into player performance and activity, facilitating more informed decision-making by coaching staff, while also providing motivation and guidance for players to improve and maintain their fitness levels.

Haversine Formula

To calculate the distance covered by the players, the Haversine formula is used. The Haversine formula is a mathematical equation used to determine the great-circle distance between two points on the surface of a sphere, such as Earth [76]. It is particularly useful when calculating distances using longitude and latitude coordinates. The formula is advantageous in this context because it accounts for the Earth's curvature, delivering more accurate distance calculations compared to simpler methods like Euclidean distance.

The Haversine formula looks like this:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \times \cos(\phi_2) \times \sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{3.1}$$

$$c = 2 \times \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = R \times c$$

Here, the latitude and longitude values should be in radians, R represents the Earth's radius (approximately 6,371 km), and atan2 is the arctangent function that returns the correct quadrant.

In this study, the Haversine formula is employed to determine the distances covered by soccer players using their GPS data. Although the curvature of the Earth may not significantly impact the calculations within the limited area of a soccer field, the use of the Haversine formula has several merits that justify its adoption in this context.

First, the Haversine formula is well suited for determining distances using longitude and latitude coordinates. As the GPS data collected from the players is inherently in this format, utilizing the Haversine formula allows for a more straightforward calculation process without the need for additional conversions. If we were to use a simpler formula, like the Pythagorean formula, additional conversion from coordinates to meters would need to be done, making the formula more complex, and negating the upside of the formula being simpler.

Second, while the Pythagorean formula may be a simpler alternative for distance calculations, it is based on the assumption that the Earth's surface is flat. This assumption can lead to inaccuracies in distance measurements. In contrast, the Haversine formula takes the Earth's curvature into account, providing more accurate distance calculations even for smaller distances like those encountered on a soccer field [76].

Moreover, using the Haversine formula can be advantageous in terms of maintaining consistency across studies. Since many research studies involving distance calculations based on GPS data employ the Haversine formula [7, 62, 73], using the same approach in this study allows for easier comparisons and potential cross-referencing with other relevant works in the field.

In conclusion, the Haversine formula is chosen for this study due to its compatibility with longitude and latitude coordinates, its ability to provide more accurate distance calculations compared to the Pythagorean formula, and its widespread use in similar research. These factors collectively contribute to the robustness and validity of the distance measurements obtained in this study.

Metabolic Power

Metabolic power is a measure of an athlete's energy expenditure during a physical activity. It is an essential parameter for understanding and analyzing the demands of various sports, especially in team sports like soccer, where players perform a wide range of movements with varying intensities. Metabolic power takes into account not only the speed and distance covered but also the intensity and energy cost of different

types of movements, such as accelerations, decelerations, and changes in direction [65].

The concept of metabolic power was introduced by Di Prampero et al. in 2005 [22], and it is calculated using the following formula:

$$P = EC * V \tag{3.2}$$

Where:
Equivalent slope $(ES) = tan(90 - \alpha)$
α = the angle between the runner and the terrain

Equivalent Mass $(EM) = g'/g$
g' = vectorial sum of forward acceleration and the acceleration of gravity
g = acceleration of gravity

Metabolic Energy $(EC) = fn(ES)xEMxKT$
KT = a fixed terrain constant of 1.29 to account for the extra energy required for the grass surface.
$fn = 155.4(ES)5 - 30.4(ES)4 - 43.3(ES)3 + 46.3(ES)2 + 19.5(ES) + 3.6$

Metabolic power is a valuable tool for assessing an athlete's energy expenditure during a game or training session. By calculating metabolic power from GPS data, coaches and sports scientists can better understand the physical demands placed on athletes and design more effective training programs and recovery strategies.

3.4.3 High Intensity Run Dataset

The high intensity run (HIR) dataset is created by us using feature extraction to provide coaches and players with valuable insights into the sprints executed by players during a match or training session. The dataset contains data about every sprint that every player makes. Features include top speed, sprint duration, average speed, and metadata like start and end positions to help facilitate visualization.

The HIR dataset has multiple potential use cases. By studying the frequency, distance, and speed of sprints, coaches and trainers assess individual player performance and identify areas for improvement. This helps in designing personalized training programs to enhance the players' fitness and skill levels [9].

Analyzing the high-intensity runs provides insights into players' fatigue levels and their ability to maintain performance throughout the match. This information is used to optimize player rotation strategies and develop more effective recovery plans [9].

By monitoring the sprint data, sports medicine professionals identify potential injury risks and address them proactively. This can include modifying training loads, implementing targeted injury prevention exercises, or making strategic substitutions during matches [9].

3.5 Data Visualization

Effective visualization tools are crucial for players, coaches, and medical staff to gain a deeper understanding of player performance, injuries, and form. To create valuable visualization tools for individuals who are not data scientists and lack experience in data analysis, it is necessary to develop visualizations that are as user-friendly as possible. Additionally, we have to prioritize the visualizations that we believe are most beneficial for the intended users.

3.5.1 Tools

GMaps

GMaps is a popular Python library that provides an interactive and user-friendly interface for visualizing geographical data using Google Maps. It allows developers to easily create maps with markers, draw shapes, display heatmaps and more [48]. By taking advantage of Google's JavaScript API for the mapping platform, this library makes integration into Jupyter notebooks or web applications much simpler.

3.5.2 Choosing What to Visualize

When choosing what to visualize and how to do it, we have two important factors to consider. The people using the visualization tools do not have a background in data science, and they have varying degrees of technical knowledge. This means that the most important factor when choosing what to visualize is how easy it is to use and read.

Many of the tools aim at coaches and other staff at the clubs. These tools mostly relate to things like team form, the distance the team runs, results compared to different objective metrics, and some visualizations to look at the correlation between subjective and objective metrics.

3.6 Injury Prediction

The goal of injury prediction is to determine if it is possible to use machine learning models and objective GPS data to predict injuries in soccer. Multiple machine learning models and data preprocessing tools are being implemented to attempt injury prediction.

3.6.1 Tools

TensorFlow

TensorFlow, developed by Google and available as open-source since 2010 [89], provides rapid deployment of machine learning models across various platforms such as CPUs, GPUs, and TPUs [26].

At its core, TensorFlow represents computations as dataflow graphs to enable efficient parallel processing and enhanced performance on large-scale data sets. It supports an array of machine learning and deep learning algorithms, neural networks, natural language processing, and computer vision applications.

TensorFlow stands out as an impressive platform thanks to its seamless integration with Keras, an intuitive high-level neural networks API designed to make building and training deep learning models simpler for both newcomers and experts. Keras makes creating and training deep learning models simple through its user-friendly interface while still providing advanced functionalities for advanced users.

Sklearn

The scikit-learn library, also referred to as sklearn, is an exceptionally popular Python library used for machine learning and data analysis. Leveraging NumPy, SciPy, and matplotlib as its foundation, it offers an impressive set of tools tailored specifically towards data science projects [33].

Sklearn provides tools for data preprocessing, such as scaling, normalization, and encoding categorical variables - essential steps towards improving model performance [82]. Furthermore, it offers methods for dimensionality reduction as well as selecting informative features to build models with.

scikit-learn offers more than just preprocessing tools; it features numerous supervised, unsupervised, and ensemble learning algorithms - making it extremely flexible in its scope of applications. Furthermore, the library includes tools for evaluating model performance such as cross-validation and various metrics that assist in selecting and comparing suitable models for comparison and selection.

Imbalanced-learn

Imbalanced-learn (imblearn) is an open source Python library created to address imbalanced datasets in machine learning [42, 52]. Imbalanced datasets, where one class significantly outnumbers another, can result in biased models that favor one over the other, leading to decreased performance on minority classes.

Imblearn provides several strategies for handling imbalanced data, such as oversampling minority classes or undersampling majority classes, or any combination thereof. It features an API compatible with scikit-learn that makes integration seamless into existing machine learning pipelines.

Imblearn offers several key techniques, including Synthetic Minority Oversampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN), Random Undersampling (RUS), and Tomek Links - each helps practitioners balance their datasets while producing more accurate and robust models for classification tasks.

Overall, imblearn is an effective tool for dealing with imbalanced datasets, providing various resampling methods which integrate seamlessly with scikit-learn and other machine learning libraries.

3.6.2 One-step Ahead Forecasting

One-step ahead forecasting, as the name implies, involves predicting the value of a single future time point. In the context of time series analysis, this is a common approach used to predict the next value in a sequence based on previous observations [68]. For instance, if you have daily measurements, like in this study, one-step ahead forecasting involves predicting the measurement for the next day.

One of the main advantages of one-step ahead forecasting is its simplicity. Since the model only needs to predict the next point in the sequence, it can often be simpler and faster to train than models that try to predict multiple points into the future. Additionally, one-step ahead forecasts can be iteratively used to create multi-step forecasts.

Injury forecasting for soccer players using GPS data is a compelling application of one-step ahead forecasting. The main objective in this context is to predict whether a player is likely to get injured in the next match or training session, based on the recent historical data of that player's physical activity.

The decision to use one-step ahead forecasting in this scenario drives by the inherent uncertainty and variability in sports injuries. Injuries are multifactorial events, influenced by a complex interplay of intrinsic (e.g., age, fitness level) and extrinsic (e.g., type of activity, environmental conditions) factors.

With one-step ahead forecasting, the model can incorporate the most recent data, which could include critical information about a player's fatigue levels, load management, and other risk factors. This is particularly important given that the risk of injury can fluctuate significantly from day to day based on these factors.

Moreover, in a practical sense, it is beneficial to predict injuries on a short-term basis (i.e., the next day) because it aligns well with the typical decision-making process in professional soccer. Coaches and medical staff often need to make daily decisions about a player's training load and participation in matches. Hence, a model that provides daily injury risk predictions can be a valuable tool in this context, aiding in the prevention and management of player injuries.

3.6.3 Creation of Training Dataset

In order to train the models for injury prediction, a dataset is created by merging the Session dataset and the injury_ts.csv file. The injury file consists of data for every player from both teams, with one row for each day over the two-year period. Each row has a corresponding 1 or 0 for every player, indicating whether they report an injury on that specific day.

The injury file contains daily data for all players over the two-year span, while the Session dataset only includes data from days when players wear the GPS tracker. For time series models, daily intervals are preferred, which leads to a discrepancy in the dataset created for model training as many days do not have matching GPS data for the injury records. The final dataset includes one row for each player on each day, containing the GPS data for that day (if a session is recorded) and the interpolated injury data for the same day. The interpolation process of the injury data is explained in Chapter 4.

3.6.4 Machine Learning Models

In this thesis, various machine learning models are implemented to predict injuries in soccer players using GPS data. These models are selected based on their ability to capture different aspects of the data and their suitability for handling time series data. The models implemented include Logistic Regression, Decision Tree, xGBoost, LSTM, GRU, and ROCKET. This section discusses the rationale behind implementing each model.

Logistic Regression

Logistic Regression is chosen as a baseline model to provide a simple, interpretable way to predict binary outcomes using GPS data. It captures the probability of a particular class or event existing, such as pass/fail, win/lose, alive/dead or injured/not injured, based on input features, offering a straightforward comparison point for more complex models.

Decision Tree

Decision Tree is implemented due to its ability to capture non-linear relationships between variables and provide interpretable decision rules.

This model can handle both continuous and categorical variables, making it a flexible choice for predicting injuries based on diverse GPS data features.

xGBoost

xGBoost, an advanced implementation of gradient boosting machines, is employed for its exceptional performance and efficiency in handling structured data. It can capture complex relationships between features and is robust to noise, making it a suitable choice for forecasting injuries using GPS data.

LSTM

Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN), are implemented due to their ability to model long-range dependencies in time series data. As GPS data is inherently sequential, LSTMs can capture temporal patterns and relationships, providing a powerful tool for injury forecasting in soccer players.

GRU

Gated Recurrent Units (GRUs), another variant of RNNs, are chosen for their simplified architecture compared to LSTMs while maintaining the ability to model long-range dependencies in time series data. GRUs can effectively capture temporal patterns in GPS data, offering an alternative approach to injury prediction with potentially lower computational costs.

ROCKET

ROCKET (RandOm Convolutional KErnel Transform), is implemented for its ability to provide highly accurate results with minimal computational requirements. By transforming the input time series data into a high-dimensional feature space using random convolutional kernels, ROCKET can efficiently capture complex patterns in GPS data, making it a promising model for injury prediction.

3.6.5 Hyperparameters

Hyperparameters play a crucial role in determining the performance of machine learning models, as they control various aspects of the learning process. To find the optimal set of hyperparameters for each model implemented in this thesis, a systematic approach using nested loops is employed.

Hyperparameter tuning involves searching through a predefined range of values to identify the combination that yields the best performance on a given task. In this thesis, multiple hyperparameters are tested for each model using nested loops. By comparing the performance of different hyperparameter combinations, the optimal configuration for each model is

determined, resulting in improved accuracy and generalization for injury prediction.

One critical hyperparameter for time series models, is the input window, which defines the length of the input sequence used to make predictions. The choice of input window size can significantly impact model performance, as it determines the amount of historical data considered by the model when making predictions. A small input window might not capture sufficient context to identify meaningful patterns, while a large input window could introduce unnecessary complexity and increase the risk of overfitting.

3.6.6 Addressing Class Imbalance

Class imbalance is a common issue in machine learning, particularly when dealing with datasets where one class significantly outnumbers the other [44]. In the context of injury prediction, the majority of data points represent non-injury instances, while injury instances are relatively rare. This imbalance can lead to biased models that prioritize the majority class, resulting in poor predictive performance for the minority class. To address class imbalance and improve the performance of the model, oversampling and undersampling techniques are employed.

Oversampling: This technique involves creating copies of instances from the minority class (injuries) to balance the class distribution. Synthetic Minority Over-sampling Technique (SMOTE) is a popular method for oversampling, which generates synthetic instances by interpolating between existing minority class samples. This approach enhances the representation of the minority class without introducing duplicates, allowing the model to better capture the patterns in the injury data [12].

Undersampling: This technique aims to balance the class distribution by randomly removing instances from the majority class (non-injuries). While this reduces the overall size of the dataset, it helps to prevent the model from being overwhelmed by the majority class. The downside of undersampling is the potential loss of valuable information due to the removal of data points. Therefore, it is essential to carefully select the instances to be removed to minimize information loss [61].

3.6.7 Training Scheme - Whole Team vs Player

Both Wiik et al. [92] and Ragab et al. [74] achieved the best results by training models on the entire team's data instead of focusing exclusively on individual players. This approach primarily addresses the issue of insufficient data, as each new player starts with no data and may have, at most, only a few hundred data points if they have been with the team for an extended period.

In our framework, we train the models using the entire team's data. This approach is anticipated to be more advantageous, particularly for the more complex deep learning models we are employing. Furthermore, since team dynamics greatly affect each player's performance, training on the entire team could help identify crucial trends. Preferably, we would compare team A and B, to see if there are any differences in the teams. But, team B does not have enough recorded injuries to be able to train models.

3.6.8 Evaluation Metrics

Accuracy

Accuracy can be determined by adding True Positive (TP) and True Negative (TN) scores and dividing by the total number of entries classified, giving an accurate representation of how many data points are classified correctly. This score gives an exact account of how many are correctly classified [40].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

Precision

Precision represents the false positive rate, meaning a model that produces no false positives has an accuracy rating of 1. This metric can be especially important in cases where false positives must be avoided and is calculated as follows [40].

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

3.6.9 Recall

Recall, also known as sensitivity or true positive rate, is a metric being used in machine learning to evaluate the performance of a classification model. It is especially important in cases where the cost of false negatives is high. Recall calculates the proportion of actual positives (TP) that are correctly identified by the model. However, a high recall does not necessarily imply a good model, as it might also have a large number of false positives. Therefore, recall is typically used in conjunction with other metrics such as precision, specificity, and the F1 score to provide a more comprehensive view of the model's performance.

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

F1-score

The F1-score is an attempt to optimize both recall and precision simultaneously, two metrics which often conflict, by creating one metric which accounts for both False Positives (FP) and False Negatives (FN). It does so

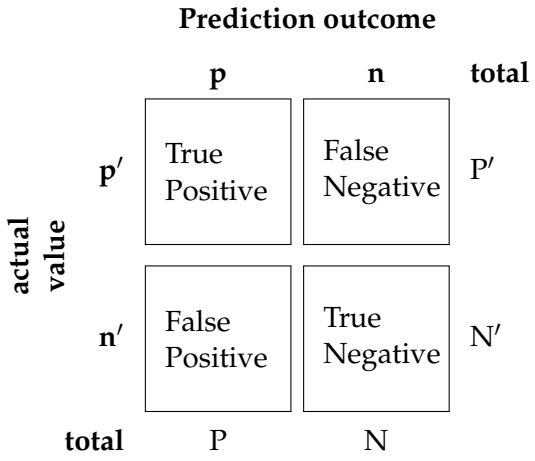


Figure 3.3: Confusion Matrix

by taking into account False Positives and False Negatives which occur simultaneously, ultimately its goal is to produce one single score which optimizes recall and precision simultaneously which would otherwise not be possible due to competing requirements of recall and precision metrics working against each other. [40]

$$F_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}} \tag{3.6}$$

3.6.10 Confusion Matrix

A confusion matrix serves as an invaluable tool for assessing the performance of classification algorithms. It presents the model's correct and incorrect predictions in a tabular format, with rows representing the actual class labels and columns indicating the predicted class labels, as seen in Figure 3.3. The main diagonal consists of correct predictions (true positives and true negatives), while off-diagonal elements signify misclassifications (false positives and false negatives). Analyzing the confusion matrix allows for evaluating classifier performance, detecting biases or imbalances, and understanding the model's error types. Used alongside other metrics like accuracy, precision, recall, and F1 score, the confusion matrix offers a comprehensive overview of a classifier's performance.

In this thesis, confusion matrices are employed to easily visualize the number of injuries correctly and incorrectly predicted. They provide a clear representation of the performance of different machine learning models, facilitating comparison and evaluation.

3.7 Chapter Summary

This chapter discusses the proposed pipeline for data import and extraction, feature extraction, visualization, and injury prediction using GPS data and machine learning in soccer players. The pipeline consists of five main steps: Data Import, Data Preprocessing, Feature Extraction, Data Visualization, and Injury Forecast.

Python libraries such as Pandas and PyArrow are employed for data import, manipulation, and processing. The Haversine formula calculates distances using GPS coordinates, while Metabolic Power offers insights into athletes' energy expenditure during physical activities.

Two new datasets, Session and High-Intensity Run (HIR), are created. The Session dataset provides an overview of each player's performance during a single session, which can be used for visualization, performance analysis, and injury prediction. The HIR dataset, on the other hand, focuses on high-intensity runs and is designed for visualization through animations, and to help analyze player performance and intensity.

This chapter also focuses on data visualization and injury prediction in soccer using various tools and techniques. For data visualization, the Python libraries GMaps and IPyWidgets are used to create user-friendly interactive visualizations. The data extraction process involves cross-checking game dates and importing subjective metrics. The visualizations are designed with non-data scientists in mind and focus on team form, results, and correlations between subjective and objective metrics.

Injury forecast involves the use of TensorFlow, scikit-learn, and imbalanced-learn libraries. The models are set up to do one-step ahead forecasting, meaning they will forecast the injury status of the next day. The datasets are created by merging the Session dataset and injury data, ensuring daily intervals are maintained. The models are trained on the entire team's data to address the issue of insufficient data and to identify trends. Evaluation metrics such as precision and F1-score are employed to assess the performance of the models for predicting injuries.

Overall, the proposed methodology aims to provide valuable insights into player performance, activity, and injury risks, enabling informed decision-making by coaching staff and facilitating player improvement. The next chapter describes the technical implementation of our pipeline.

Chapter 4

Implementation

In this chapter, we look at the implementation of our pipeline. Firstly, the preprocessing steps we take to remove erroneous data from the SoccerMon dataset are explained. Then an explanation of how we extract features from the cleaned dataset. Then we explain how we implemented satellite imagery, game results, and subjective data into our visualization. Lastly, we explain the preprocessing steps applied before injury prediction, including injury interpolation. Then we explain how we implement class imbalance handling.

4.1 Feature Extraction

We begin by importing data from a large set of files containing GPS, accelerometer, and gyroscope data, followed by cleaning up and filtering the data to remove any inconsistencies. Once we preprocess the data, we compute various features for the Session and HIR datasets, including high intensity runs, duration, total distance, average running speed, top speed, and metabolic power. Furthermore, we discuss the implementation of the Haversine formula and the Metabolic Power formula. Finally, to make the datasets easily accessible and usable for further analysis, visualization, and machine learning models, we upload the datasets to two different tables in a MySQL database.

4.1.1 Import data from CSV files

Each file in the objective dataset contains all the data captured from the GPS, accelerometer, and gyroscopes for a specific session. Each file includes the data as depicted in Table 4.1.

We download the objective folder stored on OSF and store it locally. The folder has a total size of 173GB. To iterate over all the files, we use the Python library pathlib. The fastest way to iterate over all the files is to create a list of all the file paths, then use a loop to iterate over all the files. To create the path list, we use the code shown in Figure 4.1.

Sensor technology	Metric	Description	Unit	Value Range(min-max)	Collection Frequency
NA	Player name	The player uuid	NA	UUIDv4	10hz
Clock	Time	Timestamp	HH:MM:SS.ms	13:45:53.80 / 15:47:03	10hz
GPS	Latitude	Latitude	Degrees (°)	00.000000 / 63.444820	10hz
GPS	Longitude	Longitude	Degrees (°)	00.000000 / 10.451715	10hz
GPS	Speed	Speed	Meters per second ($m.s^{-1}$)	0.000 / 6.730	10hz
Heart rate monitor	Heart rate	Heart rate	Beats per minute ($b.m^{-1}$)	0.000 / 199.000	10hz
GPS	Horizontal accuracy	Horizontal accuracy	Horizontal geometric dilution of precision (GDOP) coefficient	0.000 / 7.000	10hz
GPS	Horizontal precision	Horizontal dilution of precision	Horizontal geometric dilution of precision (GDOP) coefficient	1.000 / 31.000	10hz
GPS	Quality of signal	Quality of the GNSS/GPS signal	dBm	0.000 / 357.000	10hz
GPS	Number of satellites	Total number of satellites used in the calculation of the unit's position	Number. More is better.	0.000 / 25.000	10hz
Accelerometer	Instantaneous acceleration impulse	Absolute acceleration	m/s^{-2}	0.000 / 5.970	10hz
Accelerometer	Acceleration X	Acceleration along X-axis	g	-10.159 / 8.562	10hz
Accelerometer	Acceleration Y	Acceleration along Y-axis	g	-12.771 / 11.369	10hz
Accelerometer	Acceleration Z	Acceleration along Z-axis	g	-9.404 / 10.560	10hz
Gyroscope	Gyroscope X	Rate of rotation along X-axis	$deg.s^{-1}$	-5.931 / 10.265	100hz
Gyroscope	Gyroscope Y	Rate of rotation along Y-axis	$deg.s^{-1}$	-3.678 / 4.825	100hz
Gyroscope	Gyroscope Z	Rate of rotation along Z-axis	$deg.s^{-1}$	-3.256 / 2.323	100hz

Table 4.1: Sensor data metrics

The code in Figure 4.2 shows the loop for importing the data. We use Pandas to read the Parquet files, with Pyarrow as the engine, as it is the fastest at reading Parquet files [53]. As seen in Table 4.1, there are differences in the frequency of the GPS, accelerometer, and gyroscope. Since the gyroscope has a frequency of 100 Hz, there are 10 rows where the GPS and accelerometer data are the same, while only the gyroscope data changes. None of the features for the new datasets use the gyroscope data. Therefore, we use only every 10th line in the Parquet file.

```
days = [f"{i:02}" for i in range(1, 32)]
months = [f"{i:02}" for i in range(1, 13)]
years = ['2020', '2021']
teams = ['TeamA', 'TeamB']

data_paths = [f"PathToObjectiveFolder/{team}/{year}"
              for team in teams for year in years]
file_paths = []
for data_path in data_paths:
    for year in years:
        for month in months:
            for day in days:
                folder_path = os.path.join(data_path,
                f"{year}-{month}", f"{year}-{month}-{day}")
                file_paths.extend(
                    [str(file_path) for file_path in
                    Path(folder_path).rglob("*.parquet")]
                )
```

Figure 4.1: Code showing how the filepath list was created.

4.1.2 Filters and Sanity Checking

Before extracting features from the raw GPS data, several preprocessing steps are applied to clean and filter the data, ensuring its quality and reliability. The primary goal of preprocessing is to eliminate incorrect or unreliable data points that could potentially compromise the results of the study.

The process of selecting appropriate values for filtering the raw GPS data is largely experimental, as there is no ground truth to compare against. To test the effects of the filters, a test is set up to calculate different features using different combinations of the filters. The raw GPS data from Team A for 2020 is used for the experiment. A selection of the results from the test is seen in Table 4.2. The first row of the table contains values where only latitude and longitude values of 0 is filtered out. We can see that the highest recorded total distance is 255,5km. Assuming that a player did not run slightly more than 6 marathons during a singe session is a fair assumption, meaning that there is significant presence of erroneous data.
The filters applied are as follows:

1. Filter out rows where latitude (lat) or longitude (lon) equals 0: For lat and lon values, the filtering criterion is straightforward. Any row with values of 0 is removed. This decision is based on the observation that a lat and lon value of 0 typically indicates a faulty GPS tracker, rendering the data point unreliable. Thus, eliminating rows with these values ensures that the dataset only includes accurate positional data.

2. Filter out rows where horizontal accuracy (hacc) is more than 3: Horizontal accuracy is measured in meters and represents the radius

```
    for parquet_path in file_paths:
        columns = [["time", "lat", "lon", "speed", "
                                        inst_acc_impulse", "
                                        player_name", "hacc", "
                                        hdop", "signal_quality"
                                        ]

        df = pd.read_parquet(parquet_path, engine="auto",
                                        columns=[columns])


        df = df.iloc[::10, :]
        df = df[(df['lat'] != 0) &
        (df['lon'] != 0) &
        (df['hacc'] < 3) &
        (df['hdop'] < 10)]# &
        (df['signal_quality'] > 100)]

        df = df.reset_index()

        vals = calc_sessions_opt(df)
        data_frame = pd.DataFrame(data=[vals],
                     columns=[
                         "Team_name",
                         "Player_name",
                         "Date",
                         "Session_Id",
                         "Duration",
                         "Total_distance",
                         "Average_running_speed",
                         "Top_speed",
                         "Metabolic_power"])

    data_frames.append(data_frame)
```

Figure 4.2: Code showing the import, filtering and dataset creation.

Horizontal Accuracy	Horizontal Dilution of Precision	Signal Quality	Max Total Distance	Min Total Distance	Max Average Running Speed	Max Metabolic Power
7	31	0	255.4722	0.0503	3.0014	5037.4466
7	5	100	39.2901	1.2833	1.7305	1567.7654
5	5	100	39.0240	1.2833	1.7490	4099.1717
5	10	100	39.0240	1.2927	1.7449	1230.7780
3	10	100	36.8384	1.2927	1.9348	2321.2462
3	5	100	36.8384	1.2833	1.9357	4099.1717
7	5	200	35.5284	1.2833	1.7305	1567.7654
7	10	200	35.5284	1.2927	1.7264	1262.2980
5	10	200	35.4611	1.2927	1.7449	1230.7780
3	20	200	35.0822	1.2746	1.9348	2321.2462
3	5	300	13.4574	0.0000	1.6916	2399.9602
7	25	300	13.4574	0.0000	1.6916	2399.9602

Table 4.2: Table showing the effects of different filters on extracted features

of the margin of error for the measurement. Horizontal accuracy is measured on a scale from 0 to 7, with lower values indicating better accuracy. To ensure that only the most accurate positional data is used, rows with hacc values higher than 3 are removed from the dataset.

Due to the nature of hacc, a smaller value is selected even though the results in Table 4.2 where hacc is higher, are plausible. Since the GPS measurements are done on a relatively small surface (soccer field), when hacc is high, it means that the reported position could be significantly off from the true position. This could lead to significant errors. For example, it might appear that a player ran outside the boundaries of the field, or it could distort the path they took across the field.

3. Filter out rows where horizontal dilution of precision (hdop) is more than 10: Horizontal Dilution of Precision is a measure of the relative position of the satellites that the GPS receiver is currently tracking. It is a factor that represents the potential error or uncertainty in a measurement due to the current satellite configuration. If the satellites are closely grouped together from the perspective of the receiver, the dilution of precision will be high, and the positional data less reliable. Hdop values range from 1 to 31, with lower values

representing better accuracy.

As seen in Table 4.2, hdop does not influence the total distance, but it does affect the running speed, and therefore also metabolic power, as it uses running speed as a variable. We decided to filter out rows where hdop is more than 10, based on the max metabolic power. The max metabolic power recorded when hdop is 5, is 4099. This is a plausible number. But when hdop is anything more than 5, the highest recorded metabolic power value is 2399. Based on the significant difference between filtering out rows where hdop is more than 5, compared to the difference when using higher values such as 10, 15, or 20, indicates that there are non-erroneous data points in the range $5 \leq \text{hdop} \leq 10$. Arguments could be made to use higher hdop values than 10, but considering that the measurements are of a relatively small surface, errors are minimized when using lower hdop values.

4. Filter out rows where GPS signal quality is less than 100: Based on the results from Table 4.2, signal quality is the biggest influencer on total distance. The signal quality data ranges from 0 to 357 dBm. We test 3 different values, 100, 200, and 300. As seen in the results table, when calculating features using signal quality in the range $0 \leq \text{signal quality} \leq 357$, multiple erroneous data points are included, and the calculated total distance is not plausible. When calculating values using the range $100 \leq \text{signal quality} \leq 357$, we get a max total distance ranging from 35km to 39km, which is high but plausible. When calculating features using the range $300 \leq \text{signal quality} \leq 357$, the maximum total distance is 13km, indicating that the filter is too stringent and that multiple non-erroneous data points have been removed. Based on these findings, we filter out all rows where signal quality is less than 100.

5. Filter out rows where speed equals 0 (only for calculating Metabolic power, and HIR): Metabolic power is a metric that estimates the energy expenditure of an athlete during physical activity, taking into account factors such as running speed and acceleration. For the purpose of calculating metabolic power, it is crucial to eliminate rows with a speed value of 0. Since metabolic power is calculated by multiplying Energy cost by running speed, metabolic power will always be 0 when speed is 0. Having a metabolic power of 0 is an irrelevant data point for use in analysis and can lead to misleading results when analyzing and creating models. It is therefore removed. When calculating HIRs, rows where speed equals 0 are also removed to increase performance.

6. Change speed if it is more than 9.5: Some data points indicate that a player is running at 11.5 m/s, which is significantly higher than the world record speed for female athletes of 9.5 m/s. The fact that

multiple data points contain the value 11.5 suggests that the GPS tracker has an upper threshold of 11.5 m/s. To address this issue, it is necessary to apply an additional filter to change any data points with speed values exceeding the realistic maximum for female athletes. By changing all speeds above 9.5m/s to 9.5m/s, instead of removing them, we ensure that we do not remove potentially valuable data points for high intensity runs, while still retaining plausible data points.

4.1.3 Session Dataset

The session dataset contains different features from a single session. To calculate duration, the first and last timestamp from the raw data are collected. The Python datetime library is used to calculate the difference and return the duration of the session. Average running speed and top speed are calculated by looking at the speed parameter in the raw data. Top speed simply stores the largest value of the speed column, while the average speed is calculated by adding up all the speed values, then dividing it by the length of the resulting dataframe. The number of HIRs value in the Session dataset is not calculated during the making of this dataset. To minimize the runtime of the algorithm, the number of HIRs is added later using the HIR dataset and SQL. The total distance metric is calculated using the Haversine formula, and metabolic power is calculated using the metabolic power formula. The resulting dataset can be seen in Table 4.3

Haversine formula

To use the Haversine formula for calculating distances based on GPS data, the steps are as follows:

1. Convert the latitude and longitude values from degrees to radians.
2. Calculate the differences in latitude and longitude between the two points.
3. Plug the values into the Haversine formula to compute the great-circle distance.
4. Repeat the process for all relevant pairs of GPS coordinates, summing the calculated distances to obtain the total distance traveled by a player.

The code implementation of the Haversine formula is shown in Figure 4.3.

Metabolic Power

To calculate metabolic power from GPS data, you need to obtain the following information for each player:

Metric	Description	Unit	Example Value
teamName	Name of team player plays for	-	teamA
playerName	The anonymised player name	-	TeamA-bcç03181-2733-45d3-abf1-f7a709c63e68
sessionId	The specific session id, made up of date and player name	-	2020-06-01-TeamA-bcc03f81-2733-45d3-abf1:17a709c63e68
duration	Duration of session	Hours	02:28:24
totalDistance	Total distance covered during the session	Kilometers	11.77
averageRunningSpeed	The average running speed of the session	meter per second	1.64
topSpeed	The top speed of the session	meter per second	7.31
metabolicPower	The mean metabolic power exerted throughout the session	Metabolic power	53.66
HIR	Number of high intensity runs throughout the session	-	9

Table 4.3: Table showing the features of the Session dataset

1. Positional data (latitude, longitude, and altitude) to determine distance and elevation changes.
2. Timestamps to calculate the time duration of each activity segment.

With this data, you can use the Haversine formula to calculate the distance between consecutive GPS points and the corresponding speed. There is no measurement of altitude in the GPS data, so altitude is set to 0. Finally, apply the metabolic power formula to estimate the energy expenditure for each activity segment. The code implementation of the metabolic power formula can be seen in Figure 4.4.

4.1.4 HIR dataset

Detecting a high-intensity run involves identifying all sprints that last more than one second with a speed exceeding 5.5 m/s. HIRs have well-defined parameters and are used for multiple different studies[11][23][9]. The process begins by initializing an HIR when a player's speed surpasses the 5.5 m/s threshold. Once an HIR is initialized, a loop is created to compute various features associated with the run (seen in Figure 4.5). They include the top speed, average speed, and the total distance covered. Total distance is calculated in the same way as in the Session dataset, using the Haversine

```
def haversine(lat1, lon1, lat2, lon2):

    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2
                                           , lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2
                                           )**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371* c
return km
```

Figure 4.3: Code showing the implementation of the Haversine formula.

```
def calculate_metabolic_power(df):
    df = df[df['speed'] != 0]
    df = df.reset_index()

    g=float(9.8)
    KT = 1.29
    for i in range(len(df.index) -1):

        af=float(df.inst_acc_impulse[i])
        if af > 0:
            EM=(af**2 / g**2 + 1)**0.5
            ES=math.tan(90-math.atan(g/af))
            EC=(155.4*ES**5 - 30.4*ES**4 - 43.3*ES**3 + 46.
                                                  3*ES**2 +
            19.5*ES + 3.6) * EM * KT
            P = EC * df.speed[i]

    return np.mean(P)
```

Figure 4.4: Code showing the implementation of the Metabolic Power formula.

```
    if current_speed > 5.5 and (i == 0 or df.speed[i - 1] < 5.5
                                ):
            time_start = df.time[i]
            lat_start, lon_start = df.lat[i], df.lon[i]
            lat_end, lon_end = lat_start, lon_start
            avg_speed, tot_dist, top_speed, counter = 0, 0, 0,
                                                      0

            for j in range(i + 1, len(df)):
                lat_end, lon_end = df.lat[j], df.lon[j]
                tot_dist += haversine(lat_end, lon_end, df.lat[
                                                    j - 1], df.lon[
                                                    j - 1])
                current_speed = df.speed[j]
                avg_speed += current_speed
                top_speed = max(top_speed, current_speed)
                counter += 1

                if current_speed <= 5.5:
                    break

            avg_speed /= counter
            time_end = df.time[j - 1]
            duration = find_duration(time_start, time_end)
```

Figure 4.5: Code showing the calculations of high intensity runs.

formula. The loop continues tracking the player's movement until their speed drops below 5.5 m/s, marking the end of the HIR.

Subsequently, the duration of the HIR is calculated to determine if it exceeds one second. If the HIR meets this criterion, several features, such as top speed, average speed, distance traveled, start and end time, and latitude and longitude coordinates for both the starting and ending positions are stored. The entire process iterates through all available data files, detecting and recording HIRs for each player. Once all HIRs are identified and saved, the dataframe is exported to a CSV file for further analysis, visualization, and storage in a database.

4.1.5 Create MySQL Database for Availability

To ensure that researchers, co-master students, and other stakeholders can efficiently use the data for visualization and machine learning models, the two resulting datasets are uploaded to two different tables in a MySQL database. The tables have the same structure as the datasets shown in Table 4.3 and 4.4

The number of HIRs column in the Session dataset is extracted from the HIR dataset using SQL. When the two datasets are converted into tables in the database, the SQL query seen in Figure 4.6 is used to create the number of HIRs column in the Session table.

Metric	Description	Unit	Example Value
date	Date of the HIR	-	2020-06-01
sessionId	The specific session id, made up of date and player name	-	2020-06-01-TeamA-bcc03f81-2733-45d3-abf1-f7a709c63e68
playerName	The anonymised player name	-	TeamA-bcc03f81-2733-45d3-abf1-f7a709c63e68
teamName	Name of team player plays for	-	TeamA
timeStart	Start time of HIR	-	14:21:35.9
timeEnd	End time of HIR	-	14:21:37.6
latStart	Players latitude at the start of the sprint	-	63.44523733
lonStart	Players longitude at the start of the sprint	-	10.45186
latEnd	Players latitude at the end of the sprint	-	63.44517933
lonEnd	Player longitude at the end of the sprint	-	10.452014
avgSpeed	The average running speed of the sprint	meter per second	5.733337944444445
totDist	Total distance of the sprint	meters	10.124661901838493
topSpeed	The top speed of the sprint	meter per second	5.958338
duration	The duration of the sprint	seconds	1.7

Table 4.4: Table showing the values in the High intensity run dataset

4.2 Data Visualization

In this section, we take a look at the implementation and preprocessing steps taken concerning the creation of visualization tools. Then, we explain what data and how the data is imported.

4.2.1 Utilizing Satellite Imagery for Visualization

The effective visualization of player positions provides a practical and intuitive way to understand the movements and patterns of soccer players. To achieve this, we use satellite imagery from Google Maps, facilitated by the GMaps library in Python. Google Maps provides high-resolution satellite imagery, which enables accurate positioning of players on the actual field where the session occurs. This level of detail offers a real-world context to the GPS data, significantly enhancing the interpretability and value of the visualizations.

To position the map accurately around the soccer field, we calculate the mean of the GPS coordinates for a specific session. This mean value serves as the central point of the map, aligning the satellite imagery with the field of play. Once we position the map correctly, we can plot player positions directly onto the satellite image. Each data point corresponds to a player's location at a specific time, providing a clear visual representation of their movements throughout the session.

```
SELECT sess.Team_Name, sess.Player_name, sess.Date,
sess.Session_Id, sess.Duration, sess.Total_distance, sess.Average_running_s
FROM mpg_pmsys.LH_session sess
LEFT JOIN mpg_pmsys.LH_HIR hir ON sess.Session_Id like hir.Session_Id
GROUP BY sess.Session_Id;

ALTER TABLE mpg_pmsys.LH_session
ADD HIR_count INT;

UPDATE mpg_pmsys.LH_session AS sess
SET HIR_count = (
SELECT COUNT(hir.Session_Id)
FROM mpg_pmsys.LH_HIR AS hir
WHERE sess.Session_Id LIKE hir.Session_Id
);
```

Figure 4.6: The SQL query used to create the number of HIRs column in the Session dataset based on occurrences in the HIR dataset.

4.2.2 Incorporation of Subjective Performance Data

The SoccerMon dataset includes self-reported performance metrics for each player, for every game. These metrics encompass overall team performance, offensive performance, and defensive performance. This subjective data offers valuable insights into a player's perception of their game performance, supplementing the objective GPS data. We incorporate these metrics into multiple visualization tools to further contextualize the objective data.

We import the performance metrics using the Pandas library. The data imports from the performance.csv file, which contains a comprehensive record of player performances. Following the import, we split the performance data into two separate lists, one for Team A and another for Team B. This division enables analysis and visualization specific to each team. Once we split the data, we sort it chronologically based on the date of each game. The resulting lists for each team contain a record of the date, player name, overall team performance, offensive performance, and defensive performance for each game.

4.2.3 Gathering Game Results

To further contextualize the GPS data, and create more interesting visualization tools, a dictionary of game results is being created. The dictionary is generated by cross-referencing the registered game dates from the performance data, with the historical term lists, which contain the official game results. A dictionary is created to store the game date, result, and whether the team loses, draws, or wins.

During the process of creating the results dictionary, it is discovered that some of the registered game dates from the performance data, are not present in the official term lists. This discrepancy could be due to players participating in games for different teams (e.g, B-team), not official games (e.g., training game) or incorrect game dates entered in the PMSys app. A closer examination of the GPS data for these missing game dates reveals a significantly lower total running distance for the entire team, as compared to the game dates found in the term lists. Consequently, these dates are excluded from the visualizations using game data.

The teams and players participating in the study are required to remain anonymous, which means that the list of results cannot be published. The confidentiality requirements ensure that no personally identifiable information can be linked to the players or teams involved in the study. Player position and game results can potentially be uniquely identifiable data, that under the agreement cannot be published, therefore, it is removed from the open source code.

4.2.4 Extraction of GPS Data from Games

To obtain the GPS data specific to game days, a systematic approach is implemented, leveraging the dates from the aforementioned results dictionary. This process ensures that the collected GPS data corresponds accurately to the dates when the games occur. A loop is constructed to iteratively traverse the Session dataset. During each iteration, the algorithm checks if the date in the Session dataset matches any date from the results dictionary. If a match is found, indicating a game day, the corresponding GPS data is extracted.

4.3 Injury Prediction

This section begins by explaining the preprocessing steps applied to the dataset before injury predicting. Then it explains the implementation of class imbalance handling, the hyperparameters used, and then how we implement the different machine learning models.

4.3.1 Preprocessing

Create dataset: To predict injuries, a dataset merges from the Session dataset, and the injury dataset. The session and injury datasets merge based on the 'Date' and 'PlayerName' columns using an outer join. This ensures that all rows from both dataframes are included in the merged dataframe, even if there is no matching row in the other dataframe. This ends up creating the dataset seen in Table 4.5. The first injury data is gathered at 01.01.2020, while the earliest available GPS data is from 01.06.2020. Because of this, the injury data from before 01.06.2020 is

removed. A consequence of this is that only team A can be used for prediction. There is only one registered injury for team B after 01.06.2020, making it impossible to predict injuries for team B.

Metric	**Description**	**Unit**	**Example value**
teamName	Name of team player plays for	-	teamA
playerName	The anonymised player name	-	TeamA-bcç03181-2733-45d3-abf1-f7a709c63e68
sessionId	The specific session id, made up of date and player name	-	2020-06-01-TeamA-bcc03f81-2733-45d3-abf1-17a709c63e68
duration	Duration of session	seconds	1023
totalDistance	Total distance covered during the session	Kilometers	11.77
AverageRunning-Speed	The average running speed of the session	meter per second	1.64
topSpeed	The top speed of the session	meter per second	7.31
metabolicPower	The mean metabolic power exerted throughout the session	Metabolic power	53.66
HIR	Number of high intensity runs throughout the session	-	9
Injured	Binary value representing if a player is injured or not (1 represents injured, 0 represents not injured)	-	0

Table 4.5: Table showing the training dataset for the machine learning models

Interpolate injury values: In our analysis of injury data, we identify inconsistencies and gaps that could affect the reliability of our findings. Some players do not consistently report their injuries on a daily basis, leading to misleading values in the dataset. In the dataset, NaN values change to 0, meaning that days they do not report if they are injured or not, show as not injured in the data. To tackle this issue, a simple rule is applied

during preprocessing of the data: if a player reports being injured both the day before and the day after a 0 value, the value changes to a 1, meaning injured. This approach helps to some extent in filling the gaps in the data. However, there is still an issue where if a player does not register an injury for multiple days in a row, one consecutive injury interprets as multiple separate ones.

Create Player Sequences: To create player sequences, a function generates sequences of the GPS data for a specific player. The function takes the input dataframe, player name, and sequence length as arguments. The player's data is extracted from the dataframe and the 'Date' and 'PlayerName' columns are dropped. The function iterates through the player's data, creating sequences of the specified length, and returns a NumPy array containing these sequences. The function splits these sequences into inputs (X) and targets (y). The inputs for each sequence are all but the last row, and the target is the 'Injured' column of the last row. This means that for each sequence of data, where length equals input window, the model is being trained to predict the 'Injured' status of a player in the next time step.

4.3.2 Implementing Class Imbalance Handling

To counteract this issue of class imbalance, various techniques such as random oversampling, random undersampling, Synthetic Minority Over-sampling Technique (SMOTE), and Adaptive Synthetic (ADASYN) sampling are implemented and tested.

Random Oversampling involves duplicating instances from the minority class to balance the class distribution. Although this method can improve the model's sensitivity towards the minority class, it may also lead to overfitting due to the exact replication of instances [61].

Random Undersampling aims to balance the class distribution by randomly eliminating instances from the majority class. While this technique can help reduce the bias towards the majority class, it may also lead to loss of potentially useful information [61].

SMOTE (Synthetic Minority Over-sampling Technique) is an oversampling approach where synthetic examples are created based on the feature space similarities between existing minority instances. Rather than simple duplication, SMOTE generates new instances that are consistent with the underlying distribution of the minority class. This technique can boost the model's performance on the minority class without risking overfitting as much as random oversampling [12].

ADASYN (Adaptive Synthetic Sampling) is an advanced oversampling technique. Similar to SMOTE, ADASYN generates synthetic instances of the minority class. However, ADASYN adapts to the underlying data distribution by generating more synthetic instances for minority

```
Input window = 4, 7, 30
Test size = 0.2, 0.4
Sample modes = "none", "oversample", "undersample", "smote", "adasyn"
Sampling ratios = 0.2, 0.4
Interpolate injuries = True, False
Learning rate (only for GRU) = 0.1, 0.01
```

Figure 4.7: Hyperparameters tested for model training.

class samples that are harder to learn, as determined by their K-nearest neighbors. This results in a more nuanced oversampling strategy that can potentially lead to better performance on complex classification tasks [34].

4.3.3 Hyperparameters

All models run using various hyperparameters to identify the top performers. They test input windows of 4, 7, and 30 days, and experiment with different sizes for the test and evaluation datasets. To address class imbalance, several techniques are implemented, and multiple sampling ratios are applied. We also tested to with and without interpolation of injury values. A comprehensive list of the tested hyperparameters can be found in Figure 4.7. Each model is assessed in nested loops to determine the optimal hyperparameter configuration. The results of each run save in a list and sort based on the F1 score. The best performers are then presented, accompanied by a range of evaluation metrics. For LSTM and GRU, the number of epochs is set to 50.

4.3.4 Implementation of the Machine Learning Models

Logistic Regression: The Logistic Regression model implements using the Scikit-learn library [33]. Before training the model, the data normalizes using Tslearn's TimeSeriesScalerMinMax function [87]. The TimeSeriesScalerMinMax function is implemented instead of the more common MinMaxScaler function, as it is especially designed for time series data. It scales time series individually, so each time series in the dataset can have its own minimum and maximum after scaling. It also retains the shape of each time series. Even though the overall range of the time series is changed, the relative distances between points remain the same. After normalization, one of the sample functions to handle class imbalance is used. The model then trains and evaluates.

Decision tree: The implementation of the decision tree model is similar to the logistic regression model. It also implements using the Scikit-learn library. The data normalizes using the TimeSeriesScalerMinMax function, samples, trains, and then evaluates. Since the model generates a binary output, thresholds do not need to implement.

xGBoost: The xGBoost model implements using the xGBoost library–[13]. As before, the data normalizes using the TimeSeriesScalerMinMax function, then samples, trains, and evaluates. When training the model, it specifies that the classifier should optimize a binary logistic loss function, meaning it trains to classify instances into one of two classes.

LSTM: The LSTM model implements using the TensorFlow library [26]. The Adam optimizer is implemented due to its unique features. It maintains separate moving averages for both gradients and squared gradients. This allows it to adjust the learning rate for each parameter individually, leading to improved optimization [96]. Binary cross-entropy utilizes as the loss function. The binary cross-entropy loss function is well suited for this type of problem because it measures the difference between the predicted probability distribution and the true binary labels. It penalizes the model more heavily for making incorrect predictions, which is important in cases where the cost of false positives and false negatives is not equal [79].

GRU: The GRU model implements using the TensorFlow Keras library [26]. This model employs Gated Recurrent Units, a type of recurrent neural network architecture that is designed to tackle the vanishing gradient problem while being more computationally efficient than LSTM networks. The Adam optimizer uses in this case as well, maintaining separate moving averages of both gradients and squared gradients to adjust the learning rate for each parameter individually. Binary cross entropy is selected as the loss function for this model.

ROCKET: The ROCKET model implements using the sktime library [55]. The implementation of the ROCKET model is relatively simple, and is similar to logistic regression and decision tree. The data first scales using the TimeSeriesScalerMinMax function. A sample function then applies to handle class imbalance, before the model trains and evaluates.

4.4 Chapter Summary

This chapter explains the implementations of the three research objectives. Firstly, it discusses the implementation of the feature extraction tool. It explains the code implemented to create the two new datasets, Session and HIR.

The section starts by explaining how the raw data is imported. First, the entire SoccerMon dataset is downloaded from OSF. Then, using nested loops, a list of every possible filename is created. Once the list is created, it can iterate over all the files, and import the data for further use.

Then, it explains the preprocessing steps applied on the data, before and after features are extracted. Data is filtered to ensure quality. Filters applied include removing rows where latitude and longitude are equal to 0, horizontal accuracy is more than 3, horizontal dilution of precision is more than 10, GPS signal quality is less than 100, speed is equal to 0 (only for calculating Metabolic power, and HIR), and where speed is more than 9.5.

After explaining the preprocessing steps, it explains how the features for the Session and HIR datasets are extracted. For the Session dataset, features like duration, average running speed, top speed, total distance, and metabolic power are calculated using various methods, such as the Haversine formula for distance and a specific metabolic power formula. The HIR dataset includes features like top speed, average speed, distance traveled, start and end time, and latitude and longitude coordinates for both the starting and ending positions.

The visualization section starts by explaining how it utilizes high-resolution satellite imagery from Google Maps, for visualizing player position data. Then, it explains how subjective performance data is imported, which includes overall team performance, offensive performance, and defensive performance.

To contextualize the GPS data, game results are cross-referenced with registered game dates from the performance data. Inconsistencies are addressed by excluding games with significantly lower total running distance, and no registered game on that day in the publicly available schedule. A dictionary is created to store the game date, result, and whether the team lost, drew, or won. It is noted, however, that due to confidentiality requirements, this list cannot be published.

For game specific GPS data, a systematic approach is used, leveraging the dates from the results dictionary. A loop iteratively traverses the Session dataset to match dates and extract corresponding GPS data. The game day GPS data is then divided into two groups: Team A and Team B for team specific analyses.

The last part of the chapter explains how it implements machine learning models to predict injuries. The data from the Session and injury datasets are merged based on 'Date' and 'PlayerName' to create a new dataset. To overcome inconsistencies in the injury data, a rule is applied to interpolate injury values. Sequences are then created for all players and for each player individually in the dataset.

To handle the issue of class imbalance, techniques such as random oversampling, random undersampling, Synthetic Minority Over-sampling Technique (SMOTE), and Adaptive Synthetic (ADASYN) sampling are implemented and tested. Each technique has its advantages and potential

drawbacks, such as possible overfitting or loss of valuable data. Hyperparameters are tested extensively for each model, with the aim of identifying the best performers based on their F1 scores. A range of input windows, dataset sizes, and sampling ratios are tested. Several machine learning models are implemented for injury prediction, including Logistic Regression, Decision Tree, xGBoost, LSTM, GRU, and ROCKET.

Chapter 5

Results

In this chapter, we look at the results obtained from our research. First, the datasets created are examined and analyzed. Then, the visualization tools are explained, and potential use cases are discussed. Then we look at the results of the injury prediction models.

5.1 Feature Extraction

In this section, we look at the resulting datasets from the implementation explained in Chapter 4. We examine the datasets using stationary and seasonality tests, then the feature correlation of the datasets is examined.

5.1.1 Stationary and Seasonality Tests

In this section, we discuss the stationary and seasonality tests conducted on the Session dataset. To use the Session dataset as time series, it was devided into team A and B, then sorted by date. Since the data is not regular in terms of getting collected on a fixed interval, it consider unequally spaced. Data collected at irregular intervals are often more complex to analyze, but works for analyzing stationarity and seasonality.

These tests are essential to understand the underlying properties of the data and inform the selection of appropriate time series models for predicting or analysis. Based on our analysis, we conclude that the data is stationary and does not exhibit significant seasonality.

Stationary Test: To test the stationarity of the data, we employ the Augmented Dickey-Fuller (ADF) test. The ADF test is a widely-used statistical test that helps determine whether a time series has a unit root, implying non-stationarity [15]. A stationary time series has constant mean, variance, and auto correlation over time, which is a critical assumption for many time series models [83].

The ADF test is applied to each of the columns in the dataset, and the resulting p-values are compared to a significance level α of 0.05. In all

cases, the p-values are less than 0.05, leading us to reject the null hypothesis of the presence of a unit root. Consequently, we conclude that the data is stationary and does not require differencing or transformation to satisfy the stationarity assumption.

Seasonality Test: To assess the presence of seasonality in the data, we use the seasonal decomposition of time series (SEATS) method. This method decomposes a time series into three main components: trend, seasonal, and residual [38]. By analyzing the seasonal component, we can determine whether the data exhibits regular and predictable fluctuations over fixed intervals, such as days, weeks, months, or years.

We apply the SEATS method to each of the columns in the dataset and visually inspect the seasonal plots, a selection og the plots can be seen in Figure 5.1. In all cases, the seasonal plots appear flat and do not show any significant repeating patterns, suggesting the absence of strong seasonality in the data. Additionally, the residual plots do not display any discernible patterns, indicating that the decomposition has successfully captured the main features of the data.
Based on the results of the Augmented Dickey-Fuller stationary test and the seasonal decomposition of time series method, we conclude that the player performance data is stationary and does not exhibit significant seasonality. This information is crucial for selecting appropriate time series models for predicting or analysis, as it indicates that the data does not require differencing or seasonal adjustments.

5.1.2 Dataset Feature Correlation

When working with machine learning and visualization, it is important to thoroughly understand and analyze the data intended for use. By grasping the relationships between specific features, it becomes possible to comprehend how and why changes in certain values impact other values. In the initial analysis, the most significant information extracts from the correlation matrix, as depicted in Figure 5.2 and Figure 5.3.

The correlation matrices generate using the Session and HIR datasets that contain all 49 players from both teams. This means that the correlation scores derive from the averages of all individual players. Utilizing the average among all players is considered a fairer representation as opposed to generating the correlation matrices based on a single player, whose values could fluctuate considerably. Lighter colors in the matrix represent a stronger correlation, while darker ones indicate a weaker correlation.

As seen in Figure 5.2, there are no strong correlations in the Session dataset. The only interesting thing to note is the lack of correlation between Metabolic power and all the other variables. This can be attributed to two factors. The first being the nature of metabolic power. Metabolic power is a measure of the energy expenditure during a session. The energy systems

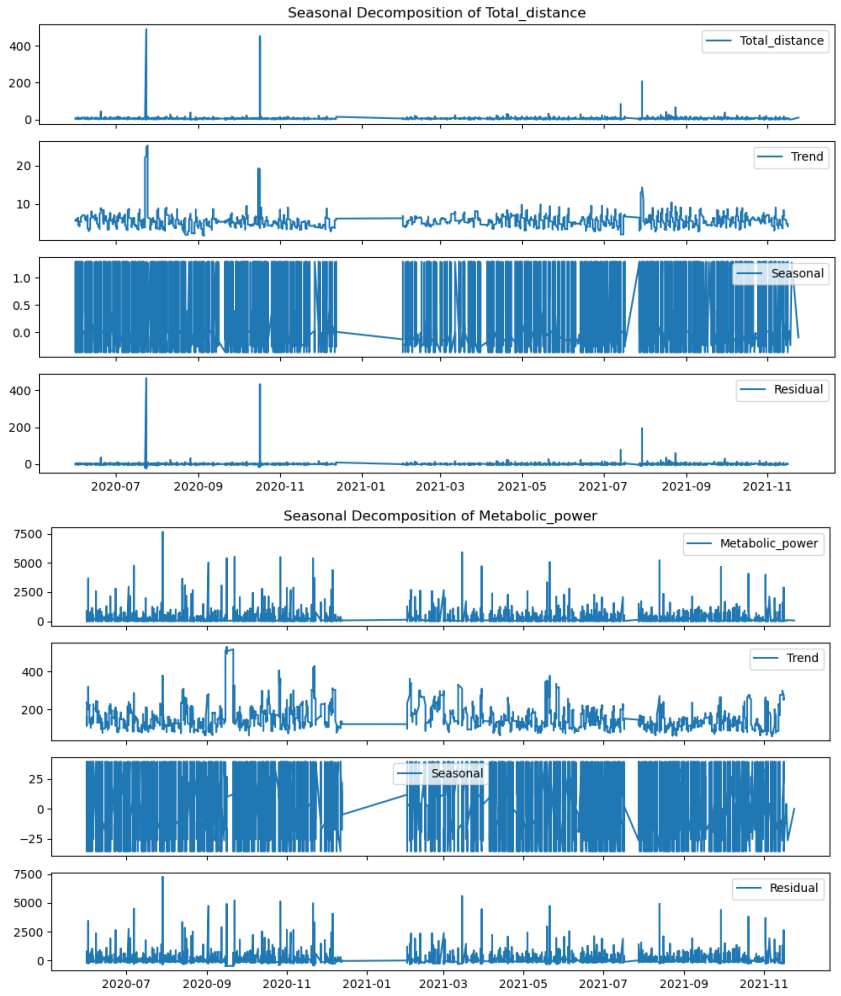


Figure 5.1: *Results of seasonality test on total distance, and metabolic power*

used during exercise can vary, depending on the intensity and duration of the activity. High-intensity short-duration activities primarily use the anaerobic energy system, while low-intensity long-duration activities rely more on the aerobic energy system.

The second factor is that metabolic power is not an independent variable, but a product of other variables in the dataset. This relationship is non-linear, which explain the low correlation values. Correlation primarily measures linear relationships, and this might be why the correlation matrix does not reflect a strong relationship between metabolic power and other variables. This mix of energy systems in use and the non-linear relationship between variables can influence the relationship between metabolic power and other variables, such as duration or distance [65]."

In Figure 5.3, we can see a clear correlation between duration and total distance. This is natural, as the longer you run over 5.5 m/s, the more distance you cover. Other than that, there are no interesting linear

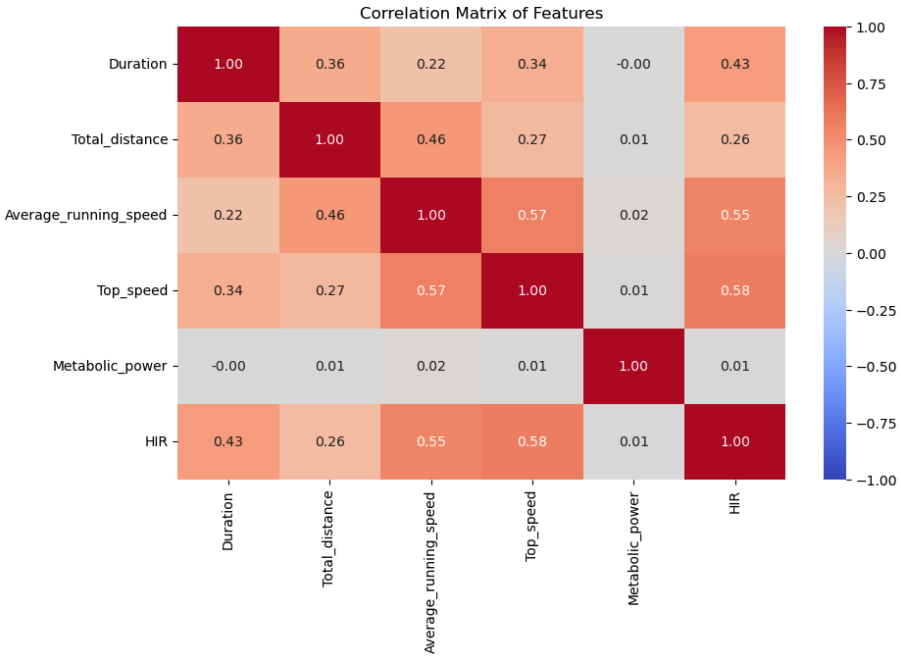


Figure 5.2: *Correlation Matrix for the Session dataset*

relationships between the variables.

5.2 Data Visualization

In this section, we provide detailed explanations for each of the visualization tools we implement. For each tool, we outline the data it utilizes, followed by a discussion about key features and potential use cases for that particular visualization.

5.2.1 Satellite Imagery

Visualizing HIRs

The first visualization we create is designed to provide a simple overview of the amount, length, and position of high intensity runs (HIRs) during a session. The plot uses data from the HIR dataset to display lines on a satellite map. The user can choose to view a single HIR or multiple HIRs at once. As seen in Figure 5.4, 5 HIRs are displayed.

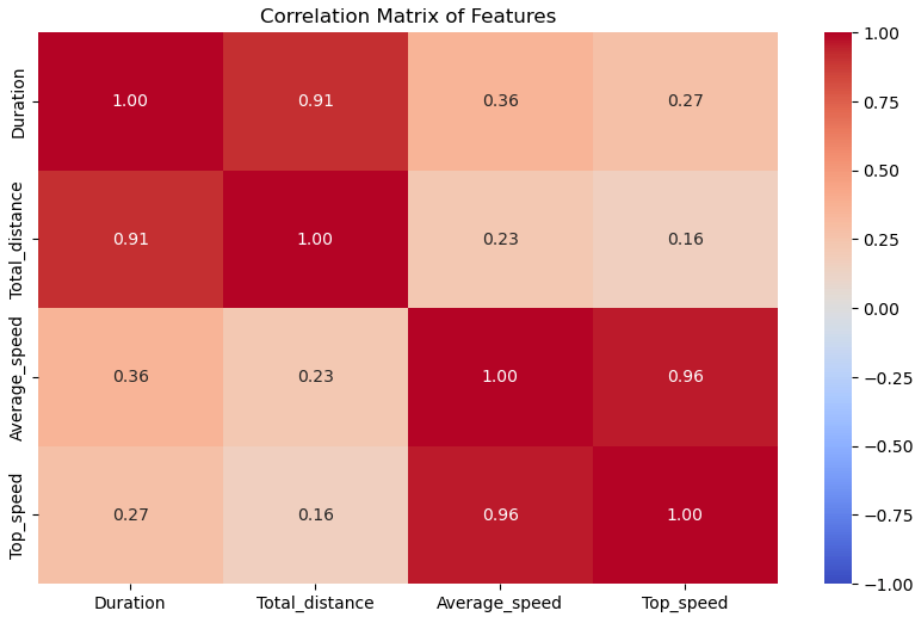


Figure 5.3: *Correlation Matrix for the HIR dataset*

Figure 5.4: *Map showing 5 different HIRs by the same player*

Key Features

1. Data Presentation: Using a satellite map for the base of the visualization gives the viewers a real-world context to the data, and it should be easy for people familiar with soccer to understand. Displaying the high-intensity runs (HIRs) as lines on this map is an intuitive way to show the length and position of each run.

2. Interactivity: The ability to select and view one or multiple HIRs is a allows users to customize the view based on their interest or needs.

Potential Use Cases

1. Player Performance: By observing the patterns and frequency of HIRs, coaches, sports scientists, and players themselves can assess a player's physical output during a game [11]. For instance, they might conclude that a player is most active during certain periods or in specific areas of the field.

2. Positioning and Tactical Analysis: Coaches can analyze a player's positioning on the field. For example, if a player is a winger and the HIRs are mostly concentrated along the sidelines, it indicates that the player is sticking to their position. Conversely, if HIRs are scattered all over the field, it might indicate a lack of positional discipline, or it could signal a tactical approach requiring the player to cover more areas.

3. Fitness and Injury Prevention: Medical staff can use this data to gauge a player's fitness and potential risk of injury [11]. If a player usually performs a high number of HIRs but this number suddenly drops, it might indicate fatigue, injury, or a lack of fitness. This data can also help to tailor individual fitness programs, ensuring each player is getting the right kind of training for their role and current physical condition.

Animating a Session

To get a better overview of a player's movement, a real-time animation of the player's movement throughout a session is created. The visualization plots a red dot on the player's position and then continuously updates as time goes on. Once per second, the dot updates its position to the next latitude-longitude pair found in the data. The visualization uses the raw GPS data from the vests. Since the SATS sport vests gather data at 100 Hz, a new dataframe is created where only one row per second is included, meaning the visualization shows how the player moved in real-time.
Key Features

1. Real-Time Animation: The visualization uses a dynamic, real-time approach to show player movement. This approach provides a continuous and intuitive representation of player movement throughout the game, which static maps or charts may not fully capture.

2. GPS Data Utilization: The tool uses raw GPS data gathered at a high frequency, providing accurate and granular data on the player's position. This high level of detail can offer valuable insights into the player's movements and positioning.

Figure 5.5: *Still image of the session animation tool*

3. User-Friendly: A red dot representing the player's position is a simple, clear visual element that most users should easily understand. It helps to make the tool user-friendly and accessible.

4. Flexibility: The tool's ability to display the player's movement throughout a session allows for flexibility. Users can choose to view the entire session or focus on specific periods of interest.

5. Compatibility with Other Tools: If used in conjunction with the previous visualization tool (satellite map showing HIRs), this tool could provide a more comprehensive understanding of a player's performance, combining both high-intensity efforts and general movement patterns.

Potential Use Cases

1. Player Movement and Tactics: The tool allows coaches and players to see how a player moves throughout a game. This could lead to insights about the player's understanding of the game, their tactical discipline, or their decision-making under pressure. For instance, a player might be drifting out of their assigned position too often or failing to make runs into space at the right time.

2. Effectiveness of Training: If the tool is used during training sessions, it could help assess the effectiveness of different drills or training methods. For example, if a training drill is designed to encourage players to make more forward runs, but the visualization shows little change in player behavior, the drill might need to be adjusted.

Heatmap

A heatmap is also created to give players and coaches a better overview of where players spent the most amount of time during a session. The visualization has two different options: it can show a heatmap for a single player, or it can show the heatmap for an entire team.

Figure 5.6: *Heatmap showing one player's session*

Key Features

1. Single Player and Team View: The tool offers the flexibility to view data for a single player or for the entire team. This allows users to analyze individual player behavior and overall team behavior, offering a range of strategic and tactical insights.
2. Player Tendencies and Movement: On a per-player basis, the heatmap can identify player tendencies and areas where the player is most active. This can help understand a player's role, their movement patterns, and whether they're adhering to tactical instructions.
3. User-Friendly Visual Representation: Heatmaps are an intuitive way to represent density data. Areas with more activity are highlighted, making it easy to see patterns and tendencies at a glance.

Potential Use Cases

1. Player Habits and Tendencies: By analyzing individual player heatmaps, coaches and analysts can identify specific areas where a player spends the majority of their time [71]. This can provide insights into player tendencies and habits, and could potentially reveal areas for improvement. For instance, a winger who tends to drift centrally might need to focus on staying wider.

2. Tactical Effectiveness: Team heatmaps can be used to evaluate the effectiveness of team tactics [71]. For example, a team trying to implement a high-pressing strategy should ideally show a high density of activity in the opposition's half. If the heatmap shows otherwise, it might indicate that the strategy isn't being implemented effectively.

3. Spatial Dominance: The tool can help in understanding which team had spatial dominance in a game. For instance, if one team's heatmap shows a high density of activity in the opposition's half, it could indicate that they had control of the game.

5.2.2 Objective Trend Diagrams

Visualizations regarding objective data is also created to help players and coaches gain more insight into a player's form and performance.

GPS Metrics
The first chart, seen in Figure 5.7, shows the average total distance run for each game. The plot can also display the total distance on a per-player basis.

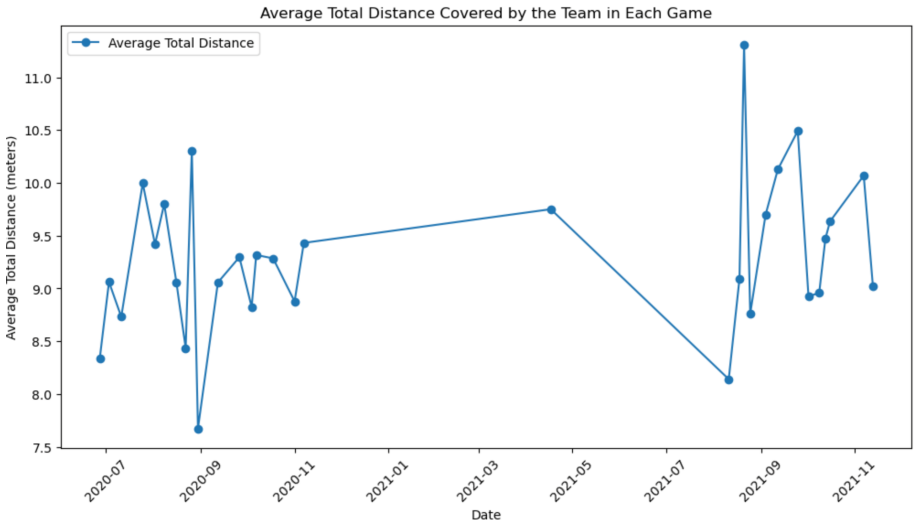


Figure 5.7: *Chart showing average total distance ran for entire team for each game.*

A chart showing the average number of High-Intensity Runs for the team on a per-game basis is also created. From the charts in Figure 5.8 and Figure ™5.7, it can be observed that the total distance run and the number of HIRs are for the most part related, meaning that, in general, games where they run longer distances also have more sprints.
The charts also share similar key features and potential use cases:
Key Features

1. Season-Long Overview: This visualization provides a holistic view of a team's physical output over an entire season. It allows for easy

comparison between matches and can highlight trends or anomalies.

2. Game-by-Game Comparison: This type of visualization allows for easy comparison between individual games. This can highlight particularly demanding matches or periods of the season.

Potential Use Cases

1. Monitoring Workload: Coaches and fitness staff can use this tool to monitor the team's workload over the season. If the average running distance is consistently high, it might indicate a risk of overtraining [51]. Conversely, a low average could suggest the team is not being pushed enough physically.

2. Tactical Analysis: The average running distance and amount of HIRs can reflect the team's tactical approach. For example, a high average might suggest a high-pressing or high-intensity style of play.

3. Preparation for Future Seasons: By looking at the running distances and amount of HIRs from a previous season, a team can better prepare for future seasons. For example, they might identify periods of the season that were particularly demanding and plan their training and player rotation strategies accordingly.

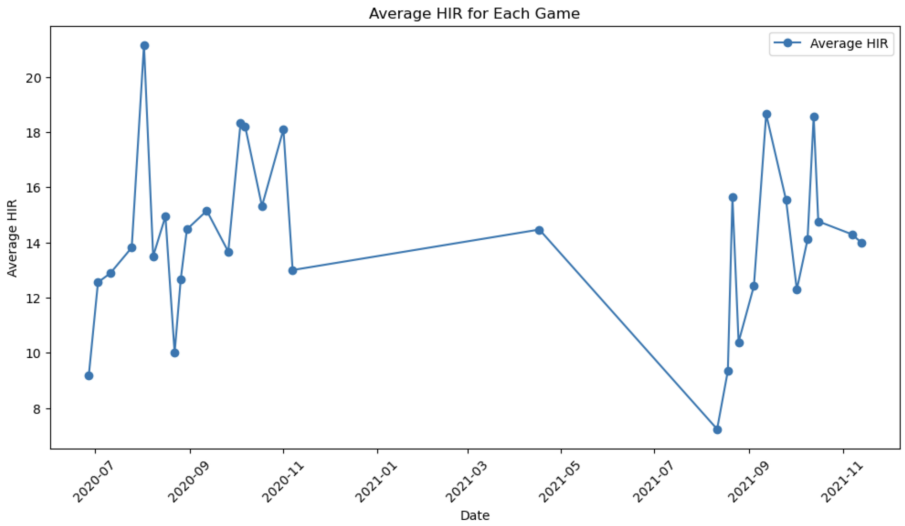


Figure 5.8: *Chart showing average number of HIRs each game*

GPS and Results

A bar chart showing the relationship between the results of games, average total distance run, and the average number of HIRs are created to potentially reveal relationships between the objective GPS data and game results. In Figure 5.9, the average distance run is shown in km. As seen in Figure 5.9, games the team lost have slightly lower average HIRs, while the

average total distance is about the same for all results. This information can help coaches and players analyze game results and potentially help them draw conclusions about what is going wrong when they lose games.

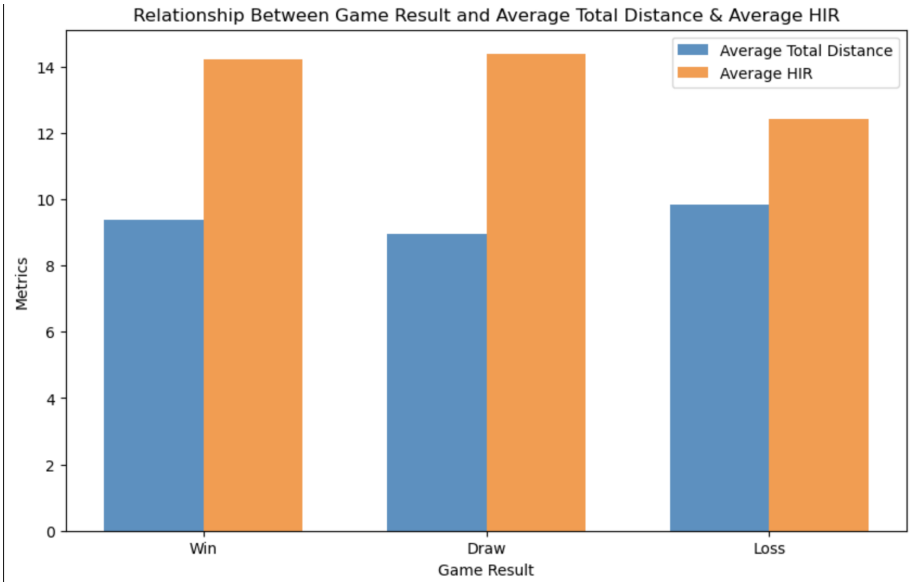


Figure 5.9: *Chart showing average distance ran and average number of HIRs grouped by results*

Key Features

1. Multifaceted Analysis: This tool incorporates multiple variables in one visualization, providing a comprehensive view of the team's performance.

2. Comparison Based on Game Results: By categorizing data based on the result of the games (win, draw, loss), this visualization allows for direct comparison of performance metrics across different outcomes.

Potential Use Cases

1. Understanding Game Performance: The visualization can help coaches and players understand the physical demands and performance during different game results. This can guide tactical and training adjustments to improve future performance.

2. Identifying Performance Factors: If the number of HIRs is consistently lower in games that the team loses, this could indicate that higher intensity running is a key factor in the team's success. The team could then focus on improving fitness levels or adjusting tactics to increase the number of HIRs.

5.2.3 Subjective Trend Diagrams

To obtain a better overview of trends, form, and performance over an extended period, multiple different visualizations are created using subjective performance data.

Subjective Performance Metrics
Firstly, a plot that displays the average subjective team performance for each game is made. The goal of the visualization is to provide coaches with an overview of the players' mentality.

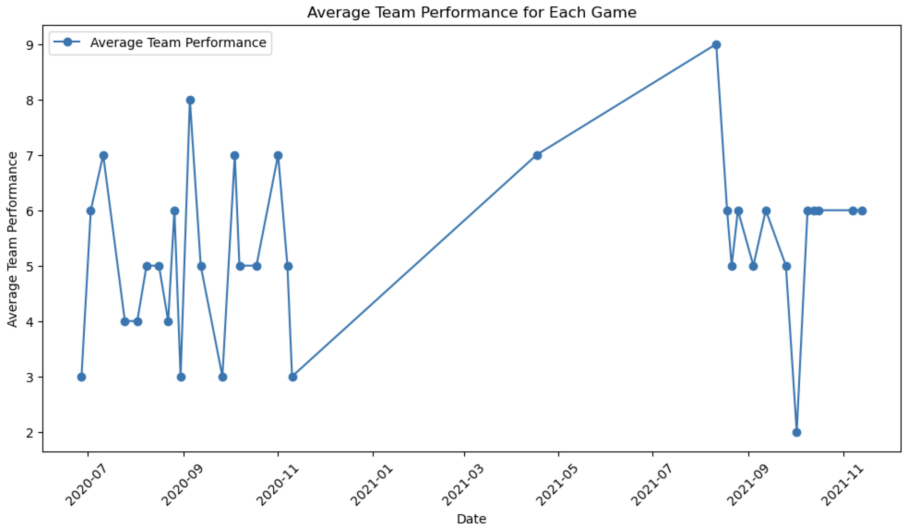


Figure 5.10: *Chart showing average subjective team performance for every game*

A more in-depth plot that displays subjective team performance, offensive performance, and defensive performance is created. Here, coaches can gain better insight into the players' mentality, and the coaches can better understand trends that might not be evident from only analyzing objective results. For example, based on Figure 5.11, the coaches can conclude that overall, over the two seasons, the players feel that the defensive performance is worse than the offensive performance. In many games, when the overall team performance was good, defensive performance was rated poor or average. Based on this, coaches can conclude that they need more work on the defensive side of the game.

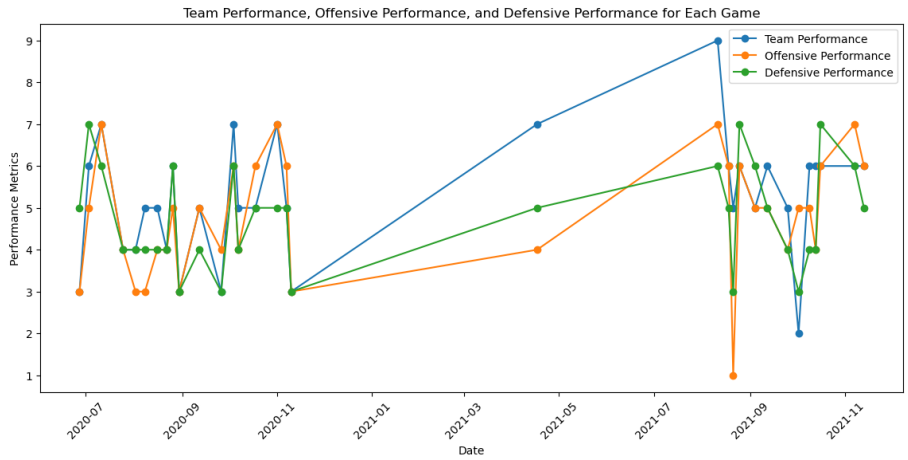


Figure 5.11: *Chart showing average overall, offensive and defensive performance for every game*

To gain even better insight into subjective game performance metrics, a bar chart showing the average team, offensive, and defensive performance for different results is created, as seen in Figure 5.12. This visualization was created with the same overall goal as the line chart in Figure 5.11, to gain better insight into the mentality of different player positions and potentially find trends that can help coaches identify issues with the team's performance.

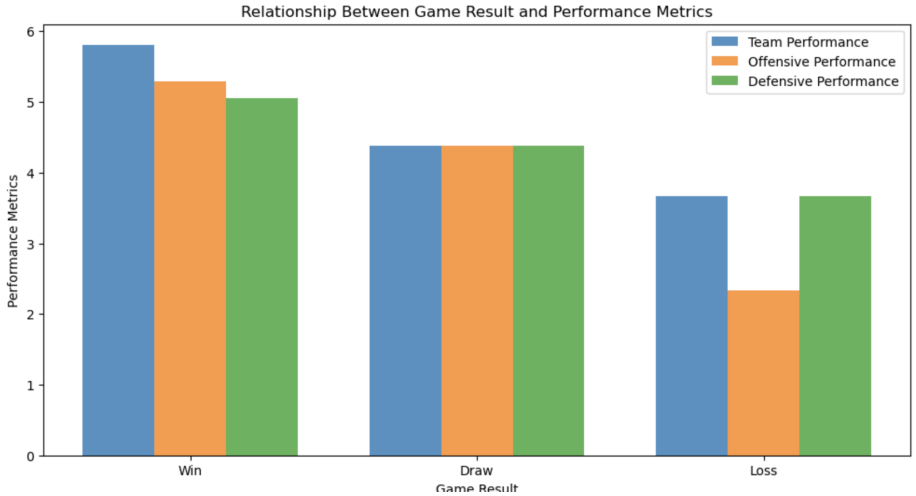


Figure 5.12: *Chart showing average overall, offensive and defensive performance grouped by results*

The plots in Figure 5.10, 5.11 and 5.12, provide a comprehensive view of the team's performance from a subjective perspective. They are designed to capture the players' perceptions of their performance, both individually and as a team. Since they have related goals, they also have related key features and potential use cases.

Key Features

1. Team, Offensive, and Defensive Performance: The plots distinguish between overall team performance and specific aspects of performance (offensive and defensive). This allows for a more nuanced understanding of the team's strengths and weaknesses.

2. Comparison Based on Game Results: The bar chart categorizes data based on the results of the games (win, draw, loss), allowing for direct comparison of subjective performance metrics across different outcomes.

Potential Use Cases

1. Understanding Team Mentality: By capturing the players' subjective assessments, these tools can provide insights into the team's mentality and morale. For example, a downward trend in subjective performance might indicate a drop in confidence or morale, which could prompt interventions to boost team spirit.

2. Identifying Tactical Issues: If players consistently rate their defensive performance as worse than their offensive performance, this could indicate tactical issues that need to be addressed in training.

Performance vs Goals

Plots showing goals scored vs. offensive team performance (Figure 5.13) and goals conceded vs. defensive performance (Figure 5.14) are created to get an even more detailed look into the mentality of the players. Here, coaches should look for significant differences in goals scored and offensive performance. If there are regularly low offensive scores while many goals are scored, it can indicate that the players lack confidence in themselves and their performance. The opposite can show that the players are overconfident. The same can be said for defensive performance and goals conceded.

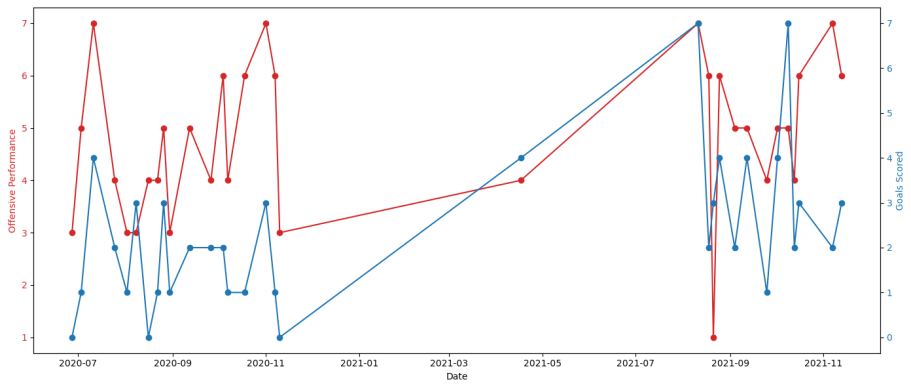


Figure 5.13: *Chart showing total number of goals scored and average offensive performance for each game.*

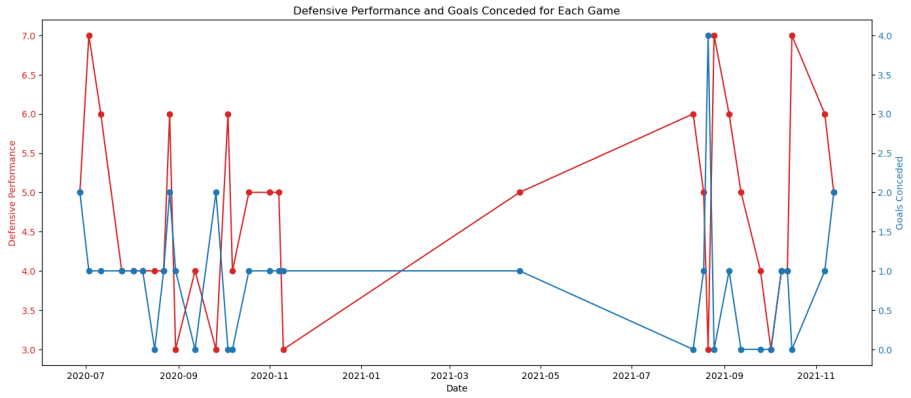


Figure 5.14: *Chart showing Goals Conceded and defensive performance.*

Key Features

1. Objective-Subjective Comparison: These plots juxtapose the subjective performance evaluation by players (offensive and defensive) against the objective measure of goals scored and conceded. This provides a unique perspective on the players' self-assessment relative to actual game outcomes.

2. Offensive and Defensive Analysis: These visualizations separate offensive and defensive performances, allowing for a detailed view of each aspect of the team's game.

3. Discrepancy Identification: These plots are specifically designed to identify significant discrepancies between subjective assessments and objective results, which can highlight issues of overconfidence or lack of self-belief.

Potential Use Cases

1. Confidence Assessment: These plots can help assess the confidence level of the team. If players rate their performance high but the team scores fewer goals or concedes more goals, it could indicate overconfidence. Conversely, if the team scores many goals or concedes few but players rate their performance low, it might indicate a lack of self-confidence.

2. Tailored Training: By understanding the discrepancies between the players' subjective performance and objective outcomes, coaches can tailor their training plans. For example, if the team scores many goals but players lack confidence in their offensive performance, coaches could incorporate activities that boost their confidence.

3. Feedback and Communication: These visualizations can serve as excellent tools for feedback and team communication. They can trigger constructive conversations about discrepancies between perception and reality, leading to a better understanding of performance and expectations.

5.3 Injury Prediction

In this section, we look at the results from the machine learning models that predict injuries in soccer. We examine multiple machine learning models, including Logistic Regression, Decision Tree, xGBoost, LSTM, GRU, and ROCKET. By evaluating each model based on their ability to accurately predict injuries, we aim to identify the most effective approach for this task. We also analyze the impact of input windows, data sampling techniques, and class imbalance on the performance of these models, providing valuable insights into the factors influencing their success.

5.3.1 Logistic Regression

The best-performing Logistic Regression algorithm correctly predicts 7 out of 11 injuries (63.63%), while incorrectly predicting 27 non-injuries as injuries as seen in Figure 5.15. The top performing logistic regression models employ the ADASYN oversampling technique, with a sampling ratio of 0.2. This results in a training dataset composed of 83% non-injuries and 17% injuries. Among the top three performers, two utilize a 7-day input window for prediction, while the best performer uses a 4-day input window. This demonstrates a clear preference for the smaller input windows of 4 and 7 days.

Input window:	4
Interpolate injuries:	False
Test Size:	0.2
Oversample Mode:	adasyn
Sampling Ratio:	0.2
Majority class size:	83.3402%
Accuracy:	0.9897%
Precision:	0.2058
Recall:	0.6363
F1:	0.3111

	Predicted	
Actual	**Injuries**	**Non-Injuries**
Actual Injuries	7	4
Actual Non-Injuries	27	2975

Figure 5.15: Logistic Regression top performer

5.3.2 Decision Tree

Figure 5.16 show that the Decision Tree model performs worse than Logistic Regression, correctly predicting 3 out of 11 injuries (28.3%), while wrongly predicting 8 non-injuries as injuries. Interestingly, the

top-performing decision tree does not employ any models to address class imbalance, resulting in the injury class comprising only 0.01% of the training dataset. The second and third best performers use random oversampling to address the class imbalance, but they do not end up getting any better predicting results, with both managing to predict 28% and 23% of the injuries. The top 3 performers all utilize a 7-day input window.

Input widow:	7
Interpolate injuries:	True
Test Size:	0.2
Oversample Mode:	none
Sampling Ratio:	0.4
Majority class size:	0.9963%
Precision:	0.3333
Recall:	0.2727
F1:	0.3

	Predicted	
Actual	**Injuries**	**Non-Injuries**
Actual Injuries	3	8
Actual Non-Injuries	6	2979

Figure 5.16: Decision Tree top performer

5.3.3 xGBoost

The xGBoost model demonstrates a clear preference for random oversampling as its method for addressing the class imbalance, as it is used in all its top performers. The same applies to the 7-day input window. All the top performers also show a strong preference for a more balanced dataset, with the minority class in the training dataset constituting 26%, compared to the 0.01% used in Logistic Regression and Decision Tree models. The xGBoost model correctly predicts 4 out of 12 injuries (25%), while wrongly predicting 6 non-injuries as injuries as seen in Figure 5.17.

5.3.4 LSTM

The LSTM model is the first to show a preference for the smaller input window of 4 days, with all of its top performers using this input window. Once again, the ADASYN method is favored for addressing class imbalance, being employed by all the top performers. From Figure5.18 we see that the best LSTM model successfully predicts 6 out of 16 injuries (37.5%), while wrongly predicting 10 non-injuries as injuries.

Input widow:	7
Interpolate injuries:	True
Test Size:	0.2
Oversample Mode:	random oversample
Sampling Ratio:	0.2
Majority class size:	74.0770%
Precision:	0.4
Recall:	0.25
F1:	0.3077

	Predicted	
Actual	**Injuries**	**Non-Injuries**
Actual Injuries	4	12
Actual Non-Injuries	6	2974

Figure 5.17: xGBoost top performer

5.3.5 GRU

The GRU model also show a preference towards using the random oversample method for handling class imbalance, using it in all of its top performers. It also prefers the 4 day input window, using it in all the top performers. The best performing GRU model correctly predicts 7 out of 16 injuries (43.75%), while wrongly predicting 13 non-injuries as injuries, as we can see in Figure 5.19.

5.3.6 ROCKET

The ROCKET model shows a preference for ADASYN, with all the top performers using it. The same can be said for the 7-day input window. Overall, the best run of the model, seen in Figure 5.20, correctly predicts 5 out of 11 injuries (45.5%), while wrongly predicting 7 non-injuries as injuries. Overall, it is by far the best performing model in terms of F1 score.

5.3.7 Discussion of Results

From the results, the best performing models in terms of F1 score is the ROCKET model. ROCKET generates features using random convolutional kernels, which are then fed into a linear classifier such as ridge regression or logistic regression. It is based on Convolutional Neural Network (CNN) technology. The main difference between ROCKET and a traditional CNN is that the convolutional kernels in ROCKET are generated randomly and are not learned from the data. Instead, the model uses these kernels to transform time series data into features that can be used by a linear classifier, such as logistic regression or ridge regression[21]. Meaning that is more light weight than traditional CNNs and RNNs like LSTM and GRU.

Input window:	4
Interpolate injuries:	True
Test Size:	0.2
Oversample Mode:	adasyn
Sampling Ratio:	0.2
Majority class size:	83.2870%
Accuracy:	0.9934
Precision:	0.375
Recall:	0.0375
F1:	0.375

	Predicted	
Actual	**Injuries**	**Non-Injuries**
Actual Injuries	6	10
Actual Non-Injuries	10	2987

Figure 5.18: LSTM top performer

This might also explain why it outperformed the more complex models of LSTM and GRU. As ROCKET uses a simple linear model for classification, it is less prone to overfitting compared to more complex models like LSTM or GRU [21].

Overall, we conclude that for this specific dataset, the ROCKET model, with a 7-day input window, is the best at predicting injuries, correctly predicting 5 out of 11 injuries, wrongly predicting 6 injuries as non-injuries, wrongly predicting 7 non-injuries as injuries, and correctly predicting 2978 non-injuries as non-injuries.

The ROCKET model is the best when it comes to F1 score, but it is also best for real-life use cases. It is much more light weight than the second best performing model, LSTM. And therefor faster and less computationally expensive to use and implement on a large scale.

Although the GRU and LSTM models are both RNNs, that aim to address the vanishing gradient problem, there are key differences that can lead to GRUs outperforming LSTMs in certain situations. GRUs may outperform LSTMs due to their simpler architecture with fewer gates, leading to reduced parameters, faster training times, and better generalization in certain situations and their computational efficiency allows for improved performance when training on long sequences or with limited resources.

Logistic Regression, Decision Trees, and xGBoost models perform relatively poorly in comparison to the LSTM, GRU and ROCKET. However, it is worth noting that the Logistic Regression model predicts the most in-

Input window:	4
Interpolate injuries:	True
Test Size:	0.2
Learning rate:	0.01
Oversample Mode:	random oversample
Sampling Ratio:	0.2
Majority class size:	74.0775%
Accuracy:	0.9926
Precision:	0.35
Recall:	0.4375
F1:	0.3889

	Predicted	
Actual	**Injuries**	**Non-Injuries**
Actual Injuries	7	9
Actual Non-Injuries	13	2984

Figure 5.19: GRU top perfromer

juries at 63% of the injuries correctly predicted as injuries, but it is also the model that wrongly predicts the most amount of injuries, meaning that the model would not be optimal in practice. The differences in performance among these models can be attributed to their inherent strengths and limitations when handling time series data. RNN-based models, such as LSTM and GRU, are generally better suited for capturing temporal dependencies in sequence data, which is essential for injury predicting [83]. In contrast, models like Logistic Regression, Decision Tree, and xGBoost might be less capable of capturing these temporal patterns, leading to reduced performance in injury prediction.

Given the success of the ROCKET model, it's worth considering that Convolutional Neural Networks (CNNs) can produce good results. Implementing a CNN, such as a Temporal Convolutional Network (TCN), might produced good results. Traditionally, CNNs are used for image and video processing, while Recurrent Neural Networks (RNNs) are more optimized for time series data [27]. In this study, RNNs were prioritized. However, for potential future work, the application of CNNs should be considered.

Looking at the results also reveals a clear preference toward the smaller input windows of 4 and 7 days. Not a single model preferred the 30-day input window, for any of its top performers. This is likely because using the 30-day input window introduces too much noise and irrelevant data. Looking back at our results, testing using more, and smaller input windows could increase performance.

Input window:	7
Interpolate injuries:	True
Test Size:	0.2
Oversample Mode:	adasyn
Sampling Ratio:	0.2
Majority class size:	83.2113%
Accuracy:	0.9956
Precision:	0.4167
Recall:	0.4545
F1:	0.4347

	Predicted	
Actual	**Injuries**	**Non-Injuries**
Actual Injuries	5	6
Actual Non-Injuries	7	2978

Figure 5.20: ROCKET top performer

Another interesting finding from the results is the discrepancy between high accuracy and a low F1 score in our model's results. This comes as a result of the imbalanced dataset. The models achieve high accuracy by simply predicting the majority class most of the time. Simply predicting not injured every time results in an accuracy of 0.99. However, this approach results in a low F1 score because the minority class is largely or completely overlooked, leading to low precision and/or recall. Since the dataset is so imbalanced, the accuracy is irrelevant for evaluating the models.

Examining the results of our study in comparison to those obtained by Rossi et al. [78], it is evident that our performance is lower. The key advantage that Rossi et al. have is a superior dataset that includes personal player data such as age, weight, and height. Additionally, their injury dataset is more comprehensive, with injuries categorized as either muscle injuries or impact injuries. Interestingly, our results show that the decision tree model is one of the worst performing models. Based on this observation, one could speculate that implementing a ROCKET model with a better dataset, similar to the one used by Rossi et al. [78], could potentially lead to the development of a model that can reliably predict injuries. This highlights the importance of having high-quality data when developing machine learning models and suggests that further improvements in injury prediction may be possible with the right data.

Looking at this from a bigger perspective, it is clear that further work needs to be done to draw any clear conclusions about the possibilities of predicting injuries using GPS data. But based on our findings, and the work done by Rossi et al., a study where a deep learning model, paired

Model	Input window	Sample mode	F1
ROCKET	7 days	ADASYN	0.4348
GRU	4 days	Random Oversample	0.3889
LSTM	4 days	ADASYN	0.375
Logistic Regression	7 days	ADASYN	0.3111
xGBoost	7 days	Random Oversample	0.3076
Decision tree	7 days	None	0.3

Table 5.1: Combined result from the top performing models, sorted based on F1 score

with a large and well-made dataset, might produce promising results.

5.4 Chapter Summary

This chapter focuses on the results of our experiments. First, the results from stationary and seasonality tests done on the Session and HIR datasets are discussed. We conclude that there is no seasonality in the datasets, and that the datasets are stationary. We then look at feature correlation for the datasets. Correlation matrices are generated from the Session and HIR datasets. It reveals no significant correlations in the session dataset, particularly between Metabolic power and other variables due to differing energy systems in use, whereas in the HIR dataset, there is a clear correlation between duration and total distance, with no other noteworthy linear relationships.

The visualization tools are then explained, for each tool, key features and potential use cases are discussed. Heatmaps and trend diagrams are used to assess player habits, tactical effectiveness, and spatial dominance based on GPS metrics. These tools also facilitate the comparison of average total distances run and High-Intensity Runs (HIRs) on a game-by-game basis, potentially highlighting overtraining or a lack of physical exertion. Objective performance data is juxtaposed with subjective performance metrics, offering insight into the players' mentality, perceived performance, and potential discrepancies between actual and perceived performance. The identification of these gaps provides opportunities for tailored training programs, improved feedback, and a better understanding of team performance and expectations.

Lastly, we examine the effectiveness of various machine learning models, such as Logistic Regression, Decision Tree, xGBoost, LSTM, GRU, and ROCKET, in predicting injuries in soccer. Each model is evaluated based on its ability to accurately predict injuries. ROCKET is found to be the best performing model. Notably, LSTM, GRU and ROCKET outperform, Logistic Regression, Decision Trees, and xGBoost, primarily due to their superior ability to capture temporal dependencies in sequence data. The study also highlights the importance of high-quality data for improving injury prediction accuracy. In conclusion, while the study provides promising insights, further research is required to make robust conclusions on injury prediction using GPS data. The next chapter discusses some of the limitations of the dataset, before we discuss our main contributions, limitation of our work and future work.

Chapter 6

Discussion

6.1 Revisiting the Problem Statement

In this section, we revisit the problem statement and research objectives defined in Chapter 1. We discuss the implementation of each research objective and assess if we effectively address the problem statement.

RO1: Effectively extract relevant features from raw GPS data to obtain features more suitable for performance and injury analysis.
Using feature extraction and engineering techniques, we effectively extract features from the raw GPS data. Before extracting the features, we clean the raw data to remove erroneous entries. We implement multiple filters: removing data points with latitude and longitude values of 0, eliminating data points with horizontal accuracy exceeding 3, horizontal dilution of precision exceeding 10, and discarding data points with GPS signal quality below 100.

The extracted features divide into two separate data sets. The Session data set contains features on a per-player-per-session basis, including Date, session ID, player name, duration, total distance, average speed, top speed, metabolic power, and number of high-intensity runs. We calculate the total distance using the haversine formula, average speed by averaging the player's speed throughout a session, and top speed by recording the highest speed value within a session. Metabolic power is calculated using the respective formula, and the number of high-intensity runs is derived from occurrences in the HIR data set.

The HIR data set consists of features on a per-sprint basis, encompassing every high-intensity run (speed > 5.5m/s, duration > 1s) from every session for each player. It includes Date, session ID, player name, latitude and longitude of the starting position, latitude and longitude of the end position, duration, distance, top speed, and average speed. We save the starting and ending position coordinates and calculate the duration using the stored timestamps for HIR initiation and completion. Total distance is determined using the Haversine formula. Top speed is recorded as

the maximum speed value, and average speed for a HIR is computed by averaging all speed values occurring within the sprint.

We successfully implement functions to effectively extract relevant features from raw GPS data. The extracted data proves to be relevant for performance and injury analysis as we address the remaining research questions.

RO2: Create visualizations using features extracted in RO1, subjective data from the SoccerMon dataset, and match results.
We create multiple visualization tools using features extracted in RO1, subjective data from the SoccerMon dataset, and match results. We develop three visualization tools that leverage satellite imagery in conjunction with data from the HIR dataset and raw GPS data. Additionally, we create four visualization tools utilizing the data from the Session dataset and match results. Lastly, we develop four different visualization tools that incorporate subjective performance data in conjunction with match results. The visualization tools created have multiple use cases and yield interesting findings.

RO3: Implement machine learning models to predict injuries based on the features extracted in RO1.
We implement six different machine learning models to predict injuries, using the Session dataset as training data. The models we employ are Logistic Regression, Decision Tree, XGBoost, LSTM, GRU, and ROCKET. The ROCKET model performs the best, with a Precision of 0.0.4167 and a Recall of 0.4545, correctly forecasting 5 out of 1 injuries. We manage to implement machine learning models that can predict injuries to a certain extent.

Can GPS data effectively analyze performance and injuries in professional female soccer?
Based on the research objectives, the answer to the research question is twofold. Can GPS data effectively analyze performance in professional female soccer? Yes, based on the findings from Objective 2 (RO2), we can conclude that GPS data is capable of analyzing performance. By combining GPS data visualization with results and subjective data, we observe a correlation between GPS data and performance. Furthermore, the visualization of GPS data offers numerous useful applications for players, coaches, and other staff members.

However, the second part of the research question, which focuses on whether GPS data can effectively analyze injuries in professional female soccer, does not yield as promising results. According to our injury forecasting models, we cannot conclude that injuries can be predicted solely based on GPS data. Nevertheless, this does not imply a complete absence of correlation between GPS data and injuries. Further research

exploring the relationship between GPS data and injuries could potentially yield interesting results.

6.2 Contributions

In this thesis, our major scientific contribution is answering the problem statement as introduced in Chapter 1. Above in Section 6.1, we elaborate on how we are able to answer this question through three research objectives. However, we also make other significant contributions such as: developing a comprehensive and adaptable feature extraction tool, making the extracted datasets publicly available, creating user-friendly visualization tools, implementing ready-to-use machine learning models with class imbalance handling, and integrating our work into the Soccer Dashboard. These achievements expand our understanding of the relationship between GPS data, subjective performance data, and injury prediction, and provide valuable resources for future researchers, players, and coaches.

All the code used in our research is publicly available through GitHub, providing other students and researchers with the opportunity to use and modify our work for their projects. The code is written to be easily modifiable and adaptable to new data or alternative purposes, encouraging collaboration and further advancements in the field. The code is published to GitHub (https://github.com/simula/pmsys).

- **Feature extraction tool:** We develop an effective feature extraction tool that can process multiple files and create new datasets. The tool is easily accessible through Jupyter Notebooks and is designed for user-friendliness. Even individuals with minimal programming knowledge can add new features or modify filters to extract data more effectively. The tool's flexibility allows for easy adjustments, such as changing the data extraction rate or implementing more stringent or lenient filters.

- **Publicly available datasets:** The datasets generated using the feature extraction tool are uploaded to a MySQL database, making them accessible to fellow master's students, researchers at Simula, and future students. These datasets have numerous potential applications, such as creating machine learning models that predict readiness or other performance-related aspects, and developing more in-depth visualization and analysis tools. They are already used to build predictive models and offer a valuable resource for future research.

- **Visualization tools:** We create a suite of visualization tools that are easy to use, even for those without prior computer or data analysis knowledge. These tools will be integrated into the Soccer Dashboard, providing players, coaches, and staff with valuable insights to inform the creation of training programs, athlete development, and mental

processes. The visualizations prove to be effective in helping users uncover information that can aid in decision-making and overall team performance.

- **Ready-to-use ML implementation with class imbalance handling:** We implement preprocessing steps and multiple machine learning algorithms for injury prediction. The publicly available Jupyter Notebook includes preprocessing steps for merging datasets, creating time series sequences, handling class imbalance using random oversampling and undersampling, SMOTE, and ADASYN, and implementing various machine learning models such as Logistic regression, decision tree, xGBoost, LSTM, GRU, and ROCKET. As the SoccerMon dataset continues to grow, the notebook can be utilized for future work on predicting injuries with larger datasets or adapted to accommodate entirely different datasets.
- **Soccer Dashboard:** The visualization tools we develop are incorporated into the Soccer Dashboard, which is currently under development. This dashboard, designed for players, coaches, and staff, will include features such as GPS playback of sessions, charts comparing various metrics, and injury prediction tools. The integration of our work into the Soccer Dashboard ensures that our research findings and tools are readily accessible to those who can benefit from them the most.

6.3 Limitations of the Dataset

In this section, we discuss some of the limitations of the SoccerMon dataset that affect the results of this study. SoccerMon is an extensive dataset that contains a vast amount of information about soccer. However, the dataset's quality, quantity, and regularity become a cause for concern. Despite the significant amount of data, there still exist gaps in the dataset that need addressing.

In this section, we delve into some of the more significant issues with the SoccerMon dataset, such as the inconsistency of data points and the lack of uniformity in data collection. We also explore how these issues impact the possibilities presented in this thesis. By addressing these concerns, we ensure that the dataset is reliable and accurate.

Moreover, it is essential to note that the availability of good data is critical in any research project. Without reliable data, it becomes challenging to draw meaningful conclusions and make informed decisions. Therefore, we must take steps to ensure that the data is of high quality and consistent. This not only benefits this paper but also has a positive impact on future research on soccer.

6.3.1 Lack of Data

The dataset contains GPS data from June 2020 until December 2021, while the subjective wellness dataset contains data from January 2020 to December 2021. The dataset is composed of data from two different teams, with a total of 49 different players appearing throughout the study. In total, the dataset contains 10750 sessions, Team A players have 5285 while Team B players have 4790 sessions. There is a wide range of how many sessions each player registers. The player with the most registered sessions has 365 sessions over 1.5 years. At the other end of the list, 3 players register with only 1 session. On average, each player registers 134 sessions over the year and a half.

A big issue with the SoccerMon dataset is that the players do not enter the subjective wellness data every day. There are a lot of subjective metrics that players are supposed to enter into the PMSys app. Some of them are on a daily basis, some after each session, and some only after each game. There is a substantial difference between how consistent the players have been in reporting in the app. Looking at a daily data point such as sleep quality, we can see that the player with the most entries has 708 out of 730 possible entries over the two years, while the player with the least entries only has 85 out of a total of 730. The mean entries for all the players are 338 of 730 possible. Not every player has been a part of the study from start to end, so expecting everyone to have filled out the daily features every day for the entire study is not feasible. But it still means that we lack about half of the values for this and all the other daily subjective features. This includes injuries.

The players do not consistently enter the per-game data either. For team A, we register a total of 33 games. Out of those 33 games, it is possible to get at least 363 data points if only 11 players register performance metrics for each game. In total, the dataset contains 136 data points. For 15 out of the 33 games, only 1 player enters subjective game performance. At the top end, there are 2 games where 13 players enter subjective game performance numbers. For team B, we register only 17 matches, where 11 of them have data from only 1 player.

To comply with data regulations, the dataset is completely anonymized. This means that the publicly available dataset lacks personal information about the players, such as names, height, weight, and age. The latter three are important factors when analyzing and predicting physiological data. The playing positions of the players are also not publicly available.

6.3.2 Lack of Separation Between Contact and Non-Contact Injuries

In the context of injury forecasting, not having a separation between contact and non-contact injuries can have a significant impact on the forecasting

models. By nature, forecasting contact injuries based on GPS data is impossible. The SoccerMon dataset does not separate between contact and non-contact injuries in its dataset, meaning that when training the models, contact injuries were included in the training data. This introduces a lot of noise in the data and will make the models perform worse.

6.3.3 COVID-19 Period

The 2020 soccer season was unlike any other due to the COVID-19 pandemic. The season was compressed, with teams playing more games in a shorter amount of time, which presented a unique challenge for players and coaches alike. This compressed season also meant that findings using data from the 2020 season may not necessarily carry over to other normal seasons [19]. As a result, it is important to carefully consider the implications of any research conducted using data from this season and to ensure that conclusions drawn are applicable to future seasons.

Another notable impact of the COVID-19 pandemic on the 2020 female soccer season was the absence of fans in the stadium. This may have affected player performance, as the lack of support and cheering from fans could impact motivation and energy levels on the field [54]. Additionally, the pandemic itself may have had long-lasting physiological effects on players, which could have affected both game and training performance. Research has shown that COVID-19 can cause a range of physiological symptoms, including respiratory issues, fatigue, and muscle weakness, which could impact athletic performance [2].

In addition to the physical effects of COVID-19, the pandemic and subsequent lockdowns may have had significant psychological impacts on players. Research has shown that lockdowns and social distancing measures can lead to increased levels of anxiety, depression, and stress, which could impact subjective wellness data, such as sleep quality, stress or fatigue [17]. These factors could have affected player motivation and overall team performance throughout the 2020 season.

6.4 Limitations of the Study

6.4.1 Limited Scope to GPS Data: Exclusion of Accelerometer and Gyroscope Data

This research confines its analysis to the GPS data extracted from the comprehensive SoccerMon dataset. However, it is worth noting that the dataset also comprises 3-Axis Accelerometer and Gyroscope data, which could potentially enable a more nuanced understanding of the athletes' physical exertion. The incorporation of these sensors could lead to the extraction of additional features, thereby enriching the dataset and facilitating a more in-depth analysis.

6.4.2 Inability to Verify the Accuracy of Extracted Features

The credibility of the extracted features remains unconfirmed due to the absence of a ground truth. The rigorous filters employed to eliminate erroneous data consequently influence the extracted features. A parameter such as total distance, for instance, could exhibit extreme variations, ranging from 13km to 250km, based on the stringency of the filters. While experimental research techniques are applied to fine tune the filters, ensuring that the extracted features are sensible, their absolute accuracy remains unverifiable.

6.4.3 Assumptions and Subjectivity in Visualization Tool Development

The development of the visualization tools is primarily guided by personal judgment due to the lack of specific information regarding the preferences of the players and coaches who utilize the PMSys system. Consequently, the visualizations created may not accurately reflect the desired parameters of these end-users.

Furthermore, the tools are not presented to players, coaches, or sports scientists for feedback, resulting in a lack of professional opinions on their utility. Consequently, the conclusions drawn from these visualizations may not align with the potential inferences of professional sports scientists, coaches, players, or medical staff.

6.4.4 Model Selection Constraints

The timeline constraints of this research limit the number and diversity of machine learning models tested. The models selected are based on several criteria, including ease of use and architectural suitability. The implemented models are chosen either for comparison with similar studies, such as the decision tree in Rossi et al. [78], or based on their potential effectiveness for injury prediction, as suggested by prior studies. Consequently, several potentially high-performing models, including but not limited to ARIMA, SARIMA, Transformer models, and Vector Autoregression, are not included in this study.

6.5 Future Work

There are several avenues for future work that can build upon the findings of this thesis. While our research makes significant strides, there remain unanswered questions and potential improvements. Some of these opportunities for future work include optimizing filters based on ground truth, creating more visualizations with different data, and using deep learning algorithms to predict injuries with larger and better datasets.

Optimize Filters Based on Ground Truth

Our research requires the implementation of filters to remove unreliable GPS data. However, the stringency of these filters can greatly influence the extracted features, such as total distance covered. Without a ground truth to compare our processed results against, we cannot confidently determine the most accurate filter parameters. Future work could involve conducting a study where players are tracked using a more reliable method than GPS, and comparing the results to data processed using different filters. This would help confirm the optimal filters to apply during the filtering process.

Extract Different Features

Expanding the scope of data analysis in future work could involve extracting different features from the SoccerMon dataset, particularly focusing on accelerometer and gyroscope data. These sensor data could provide a wealth of information about players' movements and physical states. Accelerometer data can be used to derive features related to the intensity, frequency, and types of movements, such as jumps, collisions, or changes in direction. Gyroscope data, on the other hand, can provide insights into the angular velocity and orientation changes of players, potentially identifying patterns linked to particular actions or events in a match. Such additional features could enhance the predictive power of the injury prediction model, as well as enrich the data available for visualizations and performance analysis.

Create More Visualization Tools

Due to the strict anonymization of player and team data, our options for creating visualization tools are limited. If additional data, such as player positions or game-specific information like expected goals, become available, more interesting and informative visualization tools could be developed. Expanding the range of data used in visualizations would provide users with more insights and help guide their decision-making.

Use Deep Learning Algorithms to Predict Injuries with Better Designed Datasets

Our dataset has certain limitations, including a lack of data and potential issues with data reliability. In future work, it would be interesting to apply the machine learning algorithms and class imbalance models used in this thesis to larger and higher-quality datasets. This would enable a more comprehensive investigation into the effectiveness of deep learning algorithms in predicting injuries and could lead to more accurate and reliable models.

Forcast Injuries using CNNs

Building upon the success of the ROCKET model, future work could explore the application of Convolutional Neural Networks (CNNs). Similar to the ROCKET model, CNNs employ convolutional layers, a feature that enables them to effectively extract both local and global patterns inherent in time series data. This capability aligns well with the demands of tasks such as injury forecasting in soccer.

One distinct advantage of CNNs over the ROCKET model is their proficiency in learning the most predictive features directly from data through the use of trainable convolutional filters. This is a different from the ROCKET approach, which leverages randomly generated kernels. As such, the inherent adaptability of CNNs may allow for the extraction of more meaningful and task-specific features, thereby leading to potential enhancements in performance [28].

Multimodal Injury Prediction

The study's injury prediction model was built solely from GPS data. Future studies could incorporate other types of data into this model, such as player medical histories, nutritional information, subjective wellness measures, and match outcomes. This might lead to more comprehensive and accurate injury prediction models.

6.6 Chapter Summary

In this chapter, we provide an overview of how we address the main research question (problem statement) through 3 research objectives, and the insights we derive in the process. We list the contributions of this thesis, also including numerous research artifacts we make available to the scientific community. We elaborate on the limitations of the SoccerMon dataset, as well as discuss the limitations of our own work, and provide suggestions on how potential future work can tackle some of these challenges.

Chapter 7

Conclusions

In this thesis, we develop a pipeline for extracting features, visualizing subjective wellness and performance data, objective GPS data and match data, as well as forecasting injuries using GPS data from two Norwegian female soccer teams. The pipeline begins with preprocessing the raw data and extracting features to create a dataset for visualization and model training. Before feature extraction, parameters establish, test, and implement to filter out erroneous data from the raw GPS dataset. After filtering, two new datasets create. The Session dataset, containing data on a per-player-per-session basis, includes information about specific sessions and relevant physical attributes such as distance run, top speed, metabolic power, and more. The second dataset, High Intensity Run, consists of every high intensity run (sprints with a speed > 5.5 m/s and duration > 1s). This dataset contains metadata about the sprints, as well as physical attributes like length, top speed, and so on. Creating these two datasets is the first step toward generating visualizations and predicting injuries.

The second part of the pipeline involves visualization tools. We use a combination of the datasets created in the first step, raw data, subjective performance data, and game data to create insightful and useful visualization tools that can be utilized by players, coaches, medical staff, and other team personnel. We develop three tools using satellite imagery: one for visualizing high-intensity runs, one for animating a player's position on the field throughout a session, and one to show heatmaps for players and teams. We also create multiple charts illustrating the relationship between subjective game performance, objective GPS metrics, and match data. The Visualization tools show potential for multiple different use cases, such as assisting coaches in managing tactics based on player positions, helping players prepare for higher intensity periods of seasons using trend chart of GPS data, or assisting in assessing the mindset of player based on subjective performance data and results data.

The final step of our pipeline is forecasting injuries using machine learning models. Several models are tested to identify the best-fitting model. Logistic regression, decision tree, xGBoost, LSTM, GRU, and ROCKET

all implement and evaluate with multiple hyperparameters, including different input windows for the time series. The results reveal that, for this specific dataset, the ROCKET model performs best, with precision of 0.4167 and recall of 0.4545, correctly predicting 5 out of 11 injuries and correctly predicting 2979 non-injuries as non-injuries, while incorrectly predicting 6 injuries as non-injuries and 7 non-injuries as injuries. Based on the result we conclude that on this dataset, we cannot consistently predict injuries using GPS data.

Bibliography

[1] Wesam Saleh A Al Attar and Mansour Abdullah Alshehri. 'A meta-analysis of meta-analyses of the effectiveness of FIFA injury prevention programs in soccer'. In: *Scandinavian Journal of Medicine & Science in Sports* 29.12 (2019), pp. 1846–1855.

[2] Yousef Alimohamadi, Mojtaba Sepandi, Maryam Taghdir and Hadiseh Hosamirudsari. 'Determine the most common clinical symptoms in COVID-19 patients: a systematic review and meta-analysis'. In: *Journal of preventive medicine and hygiene* 61.3 (2020), E304.

[3] Jassim Almulla, Abdulrahman Takiddin and Mowafa Househ. 'The use of technology in tracking soccer players' health performance: A scoping review'. In: *BMC Medical Informatics and Decision Making* 20 (2020), pp. 1–10.

[4] Martin Anthony, Peter L Bartlett, Peter L Bartlett et al. *Neural network learning: Theoretical foundations*. Vol. 9. cambridge university press Cambridge, 1999.

[5] Apache Software Foundation. *Apache Arrow*. https://github.com/apache/arrow. Accessed on: April 19, 2023. 2022.

[6] Horace B Barlow. 'Unsupervised learning'. In: *Neural computation* 1.3 (1989), pp. 295–311.

[7] M Basyir, M Nasir, Suryati Suryati and Widdha Mellyssa. 'Determination of nearest emergency service office using haversine formula based on android platform'. In: *EMITTER International Journal of Engineering Technology* 5.2 (2017), pp. 270–278.

[8] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella et al. 'Deep learning for time series forecasting: Tutorial and literature survey'. In: *ACM Computing Surveys* 55.6 (2022), pp. 1–36.

[9] Paul S Bradley, William Sheldon, Blake Wooster, Peter Olsen, Paul Boanas and Peter Krustrup. 'High-intensity running in English FA Premier League soccer matches'. In: *Journal of sports sciences* 27.2 (2009), pp. 159–168.

[10] Leo Breiman. 'Random forests'. In: *Machine learning* 45 (2001), pp. 5–32.

[11] Christopher Carling, Franck Le Gall and Gregory Dupont. 'Analysis of repeated high-intensity running performance in professional soccer'. In: *Journal of sports sciences* 30.4 (2012), pp. 325–336.

[12] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall and W Philip Kegelmeyer. 'SMOTE: synthetic minority over-sampling technique'. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[13] Tianqi Chen and Carlos Guestrin. 'Xgboost: A scalable tree boosting system'. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[14] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou et al. 'Xgboost: extreme gradient boosting'. In: *R package version 0.4-2* 1.4 (2015), pp. 1–4.

[15] Yin-Wong Cheung and Kon S Lai. 'Lag order and critical values of the augmented Dickey–Fuller test'. In: *Journal of Business & Economic Statistics* 13.3 (1995), pp. 277–280.

[16] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio. 'Empirical evaluation of gated recurrent neural networks on sequence modeling'. In: *arXiv preprint arXiv:1412.3555* (2014).

[17] Walter Cullen, Gautam Gulati and Brendan D Kelly. 'Mental health in the COVID-19 pandemic'. In: *QJM: An International Journal of Medicine* 113.5 (2020), pp. 311–312.

[18] Pádraig Cunningham, Matthieu Cord and Sarah Jane Delany. 'Supervised learning'. In: *Machine learning techniques for multimedia: case studies on organization and retrieval* (2008), pp. 21–49.

[19] Marc Dauty, Pierre Menu and Alban Fouasson-Chailloux. 'Effects of the COVID-19 confinement period on physical conditions in young elite soccer players.' In: *The Journal of sports medicine and physical fitness* 61.9 (2020), pp. 1252–1257.

[20] *Decision Trees*. Accessed: 2023-04-14. 2017. URL: https://www.xoriant.com/sites/default/files/uploads/2017/08/Decision-Trees-modified-1.png.

[21] Angus Dempster, François Petitjean and Geoffrey I Webb. 'ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels'. In: *Data Mining and Knowledge Discovery* 34.5 (2020), pp. 1454–1495.

[22] PE Di Prampero, S Fusi, Luigino Sepulcri, Jean-Benoıt Morin, Alain Belli and Guglielmo Antonutto. 'Sprint running: a new energetic approach'. In: *Journal of experimental Biology* 208.14 (2005), pp. 2809–2816.

[23] Valter Di Salvo, Warren Gregson, Greg Atkinson, P Tordoff and Barry Drust. 'Analysis of high intensity activity in Premier League soccer'. In: *International journal of sports medicine* 30.03 (2009), pp. 205–212.

[24] Per K Enge. 'The global positioning system: Signals, measurements, and performance'. In: *International Journal of Wireless Information Networks* 1 (1994), pp. 83–105.

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

[26] Peter Goldsborough. 'A tour of tensorflow'. In: *arXiv preprint arXiv:1610.01178* (2016).

[27] Wolfgang Groß, Sascha Lange, Joschka Bödecker and Manuel Blum. 'Predicting Time Series with Space-Time Convolutional and Recurrent Neural Networks.' In: *ESANN*. 2017.

[28] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai et al. 'Recent advances in convolutional neural networks'. In: *Pattern recognition* 77 (2018), pp. 354–377.

[29] Yonghao Gu, Yongfei Wang, Zhen Yang, Fei Xiong and Yimu Gao. 'Multiple-features-based semisupervised clustering DDoS detection method'. In: *Mathematical Problems in Engineering* 2017 (2017).

[30] Prashant Gupta. *Decision trees in machine learning*. Nov. 2017. URL: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

[31] Kevin Gurney. *An introduction to neural networks*. CRC press, 1997.

[32] Shona L Halson. 'Monitoring training load to understand fatigue in athletes'. In: *Sports medicine* 44.2 (2014), pp. 139–147.

[33] Jiangang Hao and Tin Kam Ho. 'Machine learning made easy: a review of scikit-learn package in python programming language'. In: *Journal of Educational and Behavioral Statistics* 44.3 (2019), pp. 348–361.

[34] Haibo He, Yang Bai, Edwardo A Garcia and Shutao Li. 'ADASYN: Adaptive synthetic sampling approach for imbalanced learning'. In: *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE. 2008, pp. 1322–1328.

[35] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt and Bernhard Scholkopf. 'Support vector machines'. In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[36] Kotaro Hirasawa, Masanao Ohbayashi, Masaru Koga and Masaaki Harada. 'Forward propagation universal learning network'. In: *Proceedings of international conference on neural networks (ICNN'96)*. Vol. 1. IEEE. 1996, pp. 353–358.

[37] Sepp Hochreiter and Jurgen Schmidhuber. 'Long short-term memory'. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[38] Catherine C Hood. 'Comparison of time series characteristics for seasonal adjustments from seats and x-12-arima'. In: *ASA proceedings, business and economic statistics section, alexandria, VA: ASA* (2002).

[39] David W Hosmer Jr, Stanley Lemeshow and Rodney X Sturdivant. *Applied logistic regression*. Vol. 398. John Wiley & Sons, 2013.

[40] Mohammad Hossin and Md Nasir Sulaiman. 'A review on evaluation metrics for data classification evaluations'. In: *International journal of data mining & knowledge management process* 5.2 (2015), p. 1.

[41] Kenneth Howe and Margaret Eisenhart. 'Standards for qualitative (and quantitative) research: A prolegomenon'. In: *Educational researcher* 19.4 (1990), pp. 2–9.

[42] imbalanced-learn developers. *imbalanced-learn: Over-sampling and under-sampling in Python*. Accessed: 2023-05-11. 2023. URL: https://imbalanced-learn.org/stable/.

[43] H Jabbar and Rafiqul Zaman Khan. 'Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)'. In: *Computer Science, Communication and Instrumentation Devices* 70 (2015), pp. 163–172.

[44] Nathalie Japkowicz and Shaju Stephen. 'The class imbalance problem: A systematic study'. In: *Intelligent data analysis* 6.5 (2002), pp. 429–449.

[45] Bahzad Taha Jijo and Adnan Mohsin Abdulazeez. 'Classification based on decision tree algorithm for machine learning'. In: *evaluation* 6 (2021), p. 7.

[46] Håvard D Johansen, Dag Johansen, Tomas Kupka, Michael A Riegler and Pål Halvorsen. 'Scalable Infrastructure for Efficient Real-Time Sports Analytics'. In: *Companion Publication of the 2020 International Conference on Multimodal Interaction*. 2020, pp. 230–234.

[47] Håvard D Johansen, Svein Arne Pettersen, Pål Halvorsen and Dag Johansen. 'Combining Video and Player Telemetry for Evidence-based Decisions in Soccer.' In: *icSPORTS*. 2013, pp. 197–205.

[48] Jupyter Gmaps Developers. *Jupyter Gmaps Documentation*. https://jupyter-gmaps.readthedocs.io/en/latest/. Accessed on: April 19, 2023. 2023.

[49] Alan B Krueger and David A Schkade. 'The reliability of subjective well-being measures'. In: *Journal of public economics* 92.8-9 (2008), pp. 1833–1845.

[50] John O Ledyard. 'of Experimental Research'. In: *The handbook of experimental economics* 111 (2020).

[51] M Lehmann, P Baumgartl, C Wiesenack, A Seidel, H Baumann, S Fischer, U Spöri, G Gendrisch, R Kaminski and J Keul. 'Training-overtraining: influence of a defined increase in training volume vs training intensity on performance, catecholamines and some metabolic parameters in experienced middle-and long-distance runners'. In: *European journal of applied physiology and occupational physiology* 64 (1992), pp. 169–177.

[52] Guillaume Lemaıtre, Fernando Nogueira and Christos K Aridas. 'Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning'. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 559–563.

[53] Geoffrey Lentner. 'Shared memory high throughput computing with apache arrow™'. In: *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*. 2019, pp. 1–2.

[54] Daniel Link and Gabriel Anzer. 'How the COVID-19 pandemic has changed the game of soccer'. In: *International Journal of Sports Medicine* (2021), pp. 83–93.

[55] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines and Franz J Király. 'sktime: A unified interface for machine learning with time series'. In: *arXiv preprint arXiv:1909.07872* (2019).

[56] Suresh Malodia, Nazrul Islam, Puneet Kaur and Amandeep Dhir. 'Why do people use Artificial Intelligence (AI)-enabled voice assistants?' In: *IEEE Transactions on Engineering Management* (2021).

[57] Shane Malone, Adam Owen, Bruno Mendes, Brian Hughes, Kieran Collins and Tim J Gabbett. 'High-speed running and sprinting as an injury risk factor in soccer: can well-developed physical qualities reduce the risk?' In: *Journal of science and medicine in sport* 21.3 (2018), pp. 257–262.

[58] Wes McKinney et al. 'pandas: a foundational Python library for data analysis and statistics'. In: *Python for high performance and scientific computing* 14.9 (2011), pp. 1–9.

[59] Cise Midoglu, Andreas Kjæreng Winther, Matthias Boeker, Susann Dahl Pettersen, Sigurd Pedersen, Nourhan Ragab, Tomas Kupka, Steven A. Hicks, Morten Bredsgaard Randers, Ramesh Jain, Håvard J. Dagenborg, Svein Arne Pettersen, Dag Johansen, Michael A. Riegler and Pål Halvorsen. *SoccerMon, A Large-Scale Multivariate Soccer Athlete Health, Performance, and Position Monitoring Dataset*. Open Science Framework (OSF). 2023. URL: https://doi.org/10.17605/OSF.IO/URYZ9.

[60] Puneet Misra and Arun Singh Yadav. 'Improving the classification accuracy using recursive feature elimination with cross-validation'. In: *Int. J. Emerg. Technol* 11.3 (2020), pp. 659–665.

[61] Roweida Mohammed, Jumanah Rawashdeh and Malak Abdullah. 'Machine learning with oversampling and undersampling techniques: overview study and experimental results'. In: *2020 11th international conference on information and communication systems (ICICS)*. IEEE. 2020, pp. 243–248.

[62] Tareq Monawar, Shafayat Bin Mahmud and Avijit Hira. 'Anti-theft vehicle tracking and regaining system with automatic police notifying using Haversine formula'. In: *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*. IEEE. 2017, pp. 775–779.

[63] *Neural Network*. Accessed: 2023-04-14. 2017. URL: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*Gh5PS4R_A5drl5ebd_gNrg@2x.png.

[64] Open Science Framework. *OsF.io*. https://osf.io/4znzp/. Accessed on: April 19, 2023. 2023.

[65] Cristian Osgnach, Stefano Poser, Riccardo Bernardini, Roberto Rinaldo and Pietro Enrico Di Prampero. 'Energy cost and metabolic power in elite soccer: a new match analysis approach'. In: *Med Sci Sports Exerc* 42.1 (2010), pp. 170–178.

[66] *Overfitting*. Accessed: 2023-04-14. 2019. URL: https://media.geeksforgeeks.org/wp-content/cdn-uploads/20190523171258/overfitting_2.png.

[67] pandas developers. *pandas: Powerful Python Data Analysis Toolkit*. Accessed: 2023-05-11. 2023. URL: https://pandas.pydata.org/.

[68] Georgia Papacharalampous, Hristos Tyralis and Demetris Koutsoyiannis. 'One-step ahead forecasting of geophysical processes within a purely statistical framework'. In: *Geoscience Letters* 5.1 (2018), pp. 1–19.

[69] Dabal Pedamonti. 'Comparison of non-linear activation functions for deep neural networks on MNIST classification task'. In: *arXiv preprint arXiv:1804.02763* (2018).

[70] Sigurd Pedersen, Dag Johansen, Andrea Casolo, Morten B Randers, Edvard H Sagelv, Boye Welde, Andreas Kjæreng Winther and Svein Arne Pettersen. 'Maximal strength, sprint, and jump performance in high-level female football players are maintained with a customized training program during the COVID-19 lockdown'. In: *Frontiers in Physiology* 12 (2021), p. 623885.

[71] Charles Perin, Romain Vuillemot and Jean-Daniel Fekete. 'SoccerStories: A kick-off for visual soccer analysis'. In: *IEEE transactions on visualization and computer graphics* 19.12 (2013), pp. 2506–2515.

[72] Svein A Pettersen, Håvard D Johansen, Ivan AM Baptista, Pål Halvorsen and Dag Johansen. 'Quantified soccer using positional data: A case study'. In: *Frontiers in physiology* 9 (2018), p. 866.

[73] Ryan Herwan Dwi Putra, Herry Sujiani and Novi Safriadi. 'Penerapan Metode Haversine Formula Pada Sistem Informasi Geografis Pengukuran Luas Tanah'. In: *Jurnal Sistem dan Teknologi Informasi (JUSTIN) Vol* 1.1 (2015).

[74] Nourhan Ragab. 'Soccer athlete performance prediction using time series analysis'. MA thesis. OsloMet-storbyuniversitetet, 2022.

[75] Robert Rein and Daniel Memmert. 'Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science'. In: *SpringerPlus* 5.1 (2016), pp. 1–13.

[76] C Carl Robusto. 'The cosine-haversine formula'. In: *The American Mathematical Monthly* 64.1 (1957), pp. 38–40.

[77] Frank Rosenblatt. 'The perceptron: a probabilistic model for information storage and organization in the brain.' In: *Psychological review* 65.6 (1958), p. 386.

[78] Alessio Rossi, Luca Pappalardo, Paolo Cintia, F Marcello Iaia, Javier Fernández and Daniel Medina. 'Effective injury forecasting in soccer with GPS training data and machine learning'. In: *PloS one* 13.7 (2018), e0201264.

[79] Usha Ruby and Vamsidhar Yendapalli. 'Binary cross entropy with deep learning technique for image classification'. In: *Int. J. Adv. Trends Comput. Sci. Eng* 9.10 (2020).

[80] Arthur L Samuel. 'Machine learning'. In: *The Technology Review* 62.1 (1959), pp. 42–45.

[81] Anna E Saw, Luana C Main and Paul B Gastin. 'Monitoring athletes through self-report: factors influencing implementation'. In: *Journal of sports science & medicine* 14.1 (2015), p. 137.

[82] scikit-learn developers. *scikit-learn: Machine Learning in Python*. Accessed: 2023-05-11. 2023. URL: https://scikit-learn.org/stable/.

[83] Alex Sherstinsky. 'Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network'. In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[84] STATSports. *STATSports: World Leading GPS Tracker Sports Performance Analysis*. Accessed: 2023-04-11. 2023. URL: https://statsports.com/.

[85] Ilya Sutskever, Oriol Vinyals and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 [cs.CL].

[86] Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya and Antti Sorjamaa. 'A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition'. In: *Expert systems with applications* 39.8 (2012), pp. 7067–7083.

[87] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar et al. 'Tslearn, a machine learning toolkit for time series data'. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 4686–4691.

[88] Dirk Tempelaar, Bart Rienties and Quan Nguyen. 'Subjective data, objective data and the role of bias in predictive modelling: Lessons from a dispositional learning analytics application'. In: *PLoS One* 15.6 (2020), e0233977.

[89] TensorFlow. *TensorFlow: An end-to-end open source machine learning platform*. Accessed: 2023-05-11. 2023. URL: https://www.tensorflow.org/.

[90] Greg Van Houdt, Carlos Mosquera and Gonzalo Nápoles. 'A review on the long short-term memory model'. In: *Artificial Intelligence Review* 53 (2020), pp. 5929–5955.

[91] Zhou Wang and Alan C Bovik. 'Mean squared error: Love it or leave it? A new look at signal fidelity measures'. In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.

[92] Theodor Wiik, Håvard D Johansen, Svein-Arne Pettersen, Ivan Baptista, Tomas Kupka, Dag Johansen, Michael Riegler and Pål Halvorsen. 'Predicting peek readiness-to-train of soccer players using long short-term memory recurrent neural networks'. In: *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*. IEEE. 2019, pp. 1–6.

[93] Rose Wiles, Graham Crow, Sue Heath and Vikki Charles. 'Anonymity and confidentiality'. In: (2008).

[94] Chris Wing. 'Monitoring athlete load: Data collection methods and practical recommendations'. In: *Strength & Conditioning Journal* 40.4 (2018), pp. 26–39.

[95] Xue Ying. 'An overview of overfitting and its solutions'. In: *Journal of physics: Conference series*. Vol. 1168. IOP Publishing. 2019, p. 022022.

[96] Zijun Zhang. 'Improved adam optimizer for deep neural networks'. In: *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*. Ieee. 2018, pp. 1–2.