Using Machine Learning to Predict Elite Female Athletes' Readiness to Play in Soccer

Mathias Menkerud Sagbakken

Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
60 credits

Department of Informatics
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2023

Using Machine Learning to Predict Elite Female Athletes' Readiness to Play in Soccer

Mathias Menkerud Sagbakken

Abstract

In today's world, both sports and technological advancements hold a key role in our society. Machine learning is becoming a more relevant tool in the industry due to the abundance of data available to analyze players and strategies. Especially soccer, the most watched sport in the world, embraces this technological shift to improve training conditions among players and provide insight into game strategies. To maximize performance in soccer, athletes push themselves to the limits of their potential. This rigorous process puts athletes at continuous risk of negative developments such as injuries and illness. Therefore, athletes and coaches coordinate and execute training based on experience and data from different monitoring tools to achieve a safer athletic progression. These tools result in large amounts of data, which can be difficult to interpret. A machine learning model can utilize these data to make predictions of an athlete's future performance and make the transition from raw data to strategy easier. These strategies can include choosing the most relevant players for important events and improving training conditions. Most attempts at using time series data to predict future performance among soccer players have been limited to male soccer using either wellness or positional data paired with a small selection of often simple machine learning models. In this thesis, we present a pipeline conducting several experiments to determine important data and model configurations when predicting readiness to play among professional female soccer players. The pipeline comprises data extraction, pre-processing and data analysis, experiments, and evaluation, where we visualize and quantitatively present results. We discover that by leveraging complex imputation for multivariate data, we reduce error by up to 16%. We present three use-cases and show their ability to generate actionable data that enhances player and team performance. Our proposed experiments and extensive data analysis show how utilizing an approach that can dynamically capture the unique response of players improves results. The methods used in this thesis further contribute to generalizing these time series analyses to other sports.

Acknowledgments

First and foremost, I would like to thank my supervisors, Pål Halvorsen, Cise Midoglu, Matthias Boeker, and Jim Tørresen, for their guidance and support. Their insights have been invaluable. I also want to extend my gratitude to my friends who have been there for the past five years through all courses, obligatory tasks, and thesis work. Finally, I would like to thank my family for always supporting my endeavors.

Contents

Abstract i

Acknowledgments ii

1 Introduction 1
- 1.1 Motivation 1
- 1.2 Problem Statement 3
- 1.3 Scope and Limitations 4
- 1.4 Research Methods 5
- 1.5 Ethical Considerations 6
- 1.6 Main Contributions 6
- 1.7 Thesis Outline 7

2 Background and Related Work 9
- 2.1 Terminology 9
- 2.2 Athlete Health and Performance Monitoring 10
  - 2.2.1 Wellness Reporting 10
  - 2.2.2 Training Load 11
  - 2.2.3 Injury 11
  - 2.2.4 Positional Data 11
- 2.3 Machine Learning Fundamentals 13
  - 2.3.1 Domain Awareness 13
  - 2.3.2 Data Collection 13
  - 2.3.3 Data Splits 13
  - 2.3.4 Supervised Learning 14
  - 2.3.5 Unsupervised Learning 14
  - 2.3.6 Overfitting 14
- 2.4 Machine Learning - Types of Algorithms 15
  - 2.4.1 Neural Networks 15
  - 2.4.2 Linear Regression 17
  - 2.4.3 Decision Tree Regressor 17
  - 2.4.4 Long Short-Term Memory (LSTM) Networks 18
  - 2.4.5 Convolutional Neural Networks 18
  - 2.4.6 Transformer 19
- 2.5 Machine Learning - Selected Algorithms 20
  - 2.5.1 eXtreme Gradient Boosting 20
  - 2.5.2 ROCKET 22

2.5.3 Temporal Fusion Transformer 23
2.5.4 Additional Models 24
2.6 SoccerMon Dataset 24
2.6.1 SoccerMon Subjective Metrics 24
2.6.2 SoccerMon Objective Metrics 26
2.6.3 PmSys Framework 26
2.7 Time Series Prediction for Soccer 29
2.7.1 Time Series Data 29
2.7.2 Predicting Peak Readiness-to-Train 29
2.7.3 Predicting readiness to train Using LSTM 30
2.7.4 Exploration of Different Time Series Models 30
2.7.5 Injury Forecasting With GPS data 31
2.8 Chapter Summary 32

3 Methodology **34**
3.1 Characteristics of Useful Forecasts 34
3.2 Proposed Pipeline 35
3.2.1 Data Importing 36
3.2.2 Data Analysis and Processing 37
3.2.3 Experiments Overview 38
3.2.4 Evaluation 43
3.3 System Specifications 43
3.4 Evaluation Metrics 44
3.5 Chapter Summary 46

4 Dataset Analysis and Preprocessing **47**
4.1 Overview and Composition 47
4.2 Stationarity Test 49
4.3 Feature Correlations 50
4.4 Data Distributions 53
4.4.1 Overview 53
4.4.2 Individual Players 55
4.4.3 Characteristics of Peak Readiness 56
4.5 Imputation 59
4.5.1 Statistics on Missing Data 59
4.5.2 Correlation 59
4.5.3 Distribution 60
4.5.4 Notable Changes to the Dataset 64
4.6 Feature Importance 64
4.7 Chapter Summary 66

5 Experiments and Results - First Iteration **67**
5.1 Regression Models 67
5.1.1 Model Parameters 67
5.1.2 Size of Input and Output Windows 70
5.1.3 Forecasting Horizons 72
5.1.4 Recursive Multi-step Versus Direct Forecasts 74
5.1.5 Multivariate Versus Univariate Data 75

5.1.6 Multivariate Data 76
5.1.7 Univariate Data 77
5.1.8 Comparison of Multivariate and Univariate Data . . 78
5.1.9 Team A versus Team B 79
5.1.10 Data Transferability 81
5.2 Classification Models 82
5.2.1 Model Parameters 83
5.2.2 Classification Experiment Results 83
5.3 Shortcomings and How We Can Improve Them 89
5.4 Chapter Summary 90

6 Experiments and Results - Second Iteration 91
6.1 Data Alterations 91
6.2 Experiments Overview 91
6.3 Regression Models, Second Iteration 92
6.3.1 Size of Input and Output Windows 92
6.3.2 Forecasting Horizons 94
6.3.3 Multivariate Versus Univariate Data 95
6.4 Classification Models, Second Iteration 98
6.5 Comparing First and Second Iteration Results 103
6.5.1 Comparing Dummy classification Results Across Iterations 103
6.5.2 Comparing Imputed and Non-Imputed Predictions . 105
6.6 Chapter Summary 107

7 Discussion 109
7.1 Daily and Session Based Time Series Intervals 109
7.2 Imputation 110
7.3 Input And Output Window Sizes 110
7.4 Multivariate Versus Univariate Data 111
7.5 Relevant Features and Data Types 111
7.6 Evaluation of Selected Models 112
7.7 Use-Cases 113
7.8 Hyperparameter Tuning 113
7.9 Ethical Considerations 114
7.10 Chapter Summary 114

8 Conclusion 115
8.1 Revisiting the Problem Statement 116
8.2 Other Contributions 117
8.3 Limitations of the Work 118
8.4 Future Work 118

9 Appendix 127
9.1 Correlation Matrices After Imputation 127
9.2 Boxplots Training on Both Teams Predicting Players From Either Team A or B 128
9.3 OLS-analysis 132

List of Figures

2.1 Model Fit 15
2.2 Example of a Neural Network 16
2.3 PMSys- point and click body silhouette for injury reporting. 27
2.4 PMSys- reporting training load. 28
2.5 PMSys- reporting wellness. 28

3.1 Illustration of our proposed pipeline 36
3.2 Image before and after using IterativeImputer 38
3.3 Illustration Showing The Process of Performing Sliding Window on Data 40
3.4 Confusion matrix 45

4.1 Correlation Matrix Team A before imputation 52
4.2 Correlation Matrix Team B before imputation 53
4.3 Histogram of readiness (Teams) 54
4.4 Histogram of readiness (Players) 56
4.5 Barplots- feature distribution when readiness is four or below and eight or higher before imputation 58
4.6 Binary Plot Showing Missing Dates For a Random Player . . 59
4.7 Histogram of readiness (Teams) after imputation 61
4.8 Barplots- feature distribution when readiness is four or below and eight or higher after imputation 63
4.9 Permutation feature importance plots 64
4.10 SHAP Plot 65

5.1 Lineplots showing how different input window values affect RMSE-Scores for different type of output window sizes . . . 71
5.2 Lineplot showing how RMSE-score increase with forecasting horizon 73
5.3 Direct and one-step ahead forecast 74
5.4 Barplots- Direct Versus One-Step Ahead comparison 75
5.5 Boxplots showing RMSE-Scores for all regression models for all players on team A using multivariate data 76
5.6 Boxplots showing RMSE-Scores for all regression models for all players on team A using univariate data 78
5.7 Two plots showing the best and worst performing players data using multivariate data compared to using univariate data 79

5.8 Comparing team A and B using all features for regression models 80
5.9 Comparing team A and B using univariate data for regression models 81
5.10 Confusion matrices- classifying original ten classes 84
5.11 Confusion matrices- classifying readiness peaks 86
5.12 Confusion matrices- classifying type of change in readiness . 88

6.1 Input window plots second iteration 93
6.2 Forecasting horizon plots second iteration 94
6.3 Boxplots showing RMSE-Scores using multivaraite data for the second iteration 95
6.4 Boxplots showing RMSE-Scores using univariate data for the second iteration 96
6.5 Confusion matrices- classifying original ten classes (Second Iteration) 99
6.6 Confusion matrices- classifying readiness peaks (Second Iteration) 100
6.7 Confusion matrices- classifying type of change in readiness (Second Iteration) 102
6.8 comparing forecast error for data with imputation and no imputation 105
6.9 Boxplots showing RMSE-Scores for all models using only actual days for imputed data 106
6.10 Comparing imputation to no imputation classifying peaks . 107

9.1 Correlation Matrix Team A After Imputation 127
9.2 Correlation Matrix Team B After Imputation 128
9.3 RMSE training on both team A and B predicting players from A (multivariate) 129
9.4 RMSE training on both team A and B predicting players from A (univariate) 130
9.5 RMSE training on both team A and B predicting players from B (multivariate) 131
9.6 RMSE training on both team A and B predicting players from B (univariate) 132
9.7 OLS analysis variance impact on model error 133
9.8 OLS analysis dummy model 133

List of Tables

2.1 Wellness metrics in the SoccerMon dataset 24
2.2 Training load metrics in the SoccerMon dataset 25

3.1 Description of notable python libraries used 43

4.1 Description of features used in the final dataset for the first iteration 49
4.2 Statistical properties of the distribution of readiness for both teams before imputation 55
4.3 Statistical properties of the distribution of readiness for both teams after imputation 61

5.1 LSTM- Hyperparameter configuration 68
5.2 XGBoost- Hyperparameter configuration 69
5.3 TFT- Hyperparameter configuration 70
5.4 Table describing the boxplots in Figure 5.5 with numerical values 77
5.5 Table describing the boxplots in Figure 5.6 with numerical values 78
5.6 Table describing the statistical properties of Team A and B RMSE-scores found in Figure 5.8 80
5.7 Table describing the statistical properties of Team A and B RMSE-scores found in Figure 5.9 81
5.8 classification metrics- results ten classes 84
5.9 classification metrics- results peaks 86
5.10 classification metrics- results readiness change 88

6.1 Table describing the statistical properties of the RMSE values found in Figure 6.3 95
6.2 Table describing the statistical properties of the RMSE values found in Figure 6.4 97
6.3 Table showing percentage decrease in loss using multivariate over univaraite data 98
6.4 classification metrics- results ten classes- sessions 99
6.5 classification metrics- results peaks 101
6.6 classification metrics- results readiness change- Sessions . . 102
6.7 Table showing accuracy score for all classification tasks across both iterations 103

6.8 Table showing F1-scores for all classification tasks across both iterations 104
6.9 Table describing the statistical properties of the RMSE values in Figure 6.9 107

Acronyms

ADF Augmented Dickey-Fuller test. 50

AI artificial intelligence. 13, 114

LPM Local Position Measurement. 12

LSTM Long short-term memory. 3, 7, 18, 24, 29, 30, 41, 67, 68, 71–73, 76, 78, 80, 82, 84, 86–89, 92, 94, 97, 98, 100, 101, 106, 110, 112, 113

ML Machine Learning. 1–3, 7, 9, 13, 15, 20, 28–30, 32, 34, 43, 50, 60, 70, 77, 103, 110, 115, 118, 119

MLP Multilayer Perceptron. 16

MSE Mean Square Error. 44

NLP Natural Language Processing. 19

RMSE Root Mean Square Error. 44, 70–74, 77, 79–81, 89, 92, 94–97, 103, 106, 108, 109, 111, 112, 116, 117, 128, 132, 133

RNN recurrent neural network. 18, 30

TFT Temporal Fusion Transformer. 23, 41, 44, 67, 69, 72, 73, 76, 78, 80–82, 91, 92, 94–97, 106, 107, 110–113

TPE Tree-based Parzen Estimators. 41, 69

Chapter 1

Introduction

Analyzing time series data using Machine Learning (ML) to find patterns and trends in the well-being of soccer players might uncover valuable insight. With such an approach, a model can be fed data from multiple monitoring devices used by athletes, such as subjective wellness feedback and objective positional data. The resulting data can be transformed into easily understandable indicators of an athlete's future performance. This knowledge, in turn, can be valuable for a coach to make decisions for important events. However, most attempts using athlete time series data in soccer to predict future performance have mainly used subjective wellness or positional data. Therefore, using relevant features from both data types motivates the prospect of better results. This thesis explores several ML and deep learning models to forecast *readiness to train*, a self-reported metric describing the perceived ability to perform among professional female soccer players. The aim is to explore different approaches to both data and model configurations to give insight into important factors when forecasting such time series.

1.1 Motivation

Soccer clubs are interested in models that analyze training data and give concrete and understandable feedback in a way that helps coaches with decision-making [33, 57]. Therefore, a ML approach must produce statistics represented in a way that can support making decisions for relevant strategies on the playing field. With this in mind, a model that can reliably predict an athlete's future well-being will benefit clubs.

Injuries are a regular occurrence in soccer and affect the injured player's performance as well as their team's [3]. The monetary cost of injuries was, among others, made apparent by Eliakim et al. [17]. They estimated the average cost of injuries among English Premier League teams for a single season was 45 million pounds. This number was calculated based on the weaker performance of a team due to the injured player combined with the injured player's salary. It would then stand to reason that preventing injuries with any available tool would be invaluable in saving costs.

Subjective wellness feedback is an essential and much-used method to track an athlete's performance [31, 36]. Knowing how well an athlete performed and overall felt during previous training sessions and competitions makes it easier to keep track of progress and how best to make improvements for further athletic development. Subjective wellness feedback is a field with plenty of research, and impactful systems that collect wellness data from athletes are already made evident by Johansen et al. [31] with the PmSys monitoring system. The wellness metric Readiness to train originates from the PmSys monitoring system [31] and measures a player's perceived ability to perform for a given session. Readiness plays a crucial part in this thesis as it will be used to map the future performances of elite female soccer players.

Utilizing readiness to predict how well a player will perform in future events offers opportunities to maximize player and team performance. By analyzing development in readiness among athletes, it is possible to extrapolate trends that can lead to enhanced performance or risk of negative development, such as injuries and overload [49, 60]. Such analyses improve training conditions among players. Further, clubs can use readiness for important events such as matches to derive strategies. These strategies can be to select a team lineup, choosing the most ready players at the most optimal times during a match. In this thesis, in addition to readiness, we will look at other wellness features but also features derived from GPS data and see whether they can complement our model predictions.

Positional data extracted using GPS or similar technology make it possible to create objective metrics such as total distance traveled, average running speed, and number of accelerations above a certain speed. Such metrics can give insight into an athlete's objective performance on the field [49]. Positional data also has the advantage that it is objective and inaccurate data is a cause of faulty equipment and technology rather than subjective reports from a participant. A recent study by Lourencco et al. [36] showed that objective and subjective fatigue monitoring data complement each other since they describe different aspects of fatigue. These findings suggest that objective and subjective data are needed to gain a holistic perspective on how fatigue affects athletes. Therefore, exploring several objective and subjective features and their effect on predicting readiness might yield better results. Further, leveraging recent algorithmic advancements using current state-of-the-art time series models could also provide an improved prediction outcome.

Using ML in soccer to forecast an athlete's future performance is a relatively new idea that has already seen promising results [4, 49, 60]. There are numerous ML models, and finding a suitable one that works optimally can prove challenging. Such a model has to handle sparse data and not be too computationally taxing as the forecasts will be in the very near future.

Long short-term memory (LSTM) [27] is a much-used model for time series forecasting and has already been successfully tested to forecast future athletic performance among soccer players [60]. However, other ML architectures such as transformers and convolutional neural networks have proven to perform as good or better in terms of accuracy and efficiency for time series forecasting [12, 18, 63]. Due to their exceptional performance with spatial data, convolutional neural networks can also effectively manage 1D time series data. The transformer [58] has gained popularity for its ability to predict larger forecasting horizons. The attention mechanism of the transformer allows for relational properties to be weighted such that the relationship between certain data points across time is more important than others, which is vital for solving time series problems.

Using an ML approach to analyze subjective wellness and positional- data might uncover promising results that would otherwise be impossible to discover by only looking at one or the other. There are already examples of wellness and positional data used in ML predictions for professional soccer players [60] [49], but to our knowledge, using both kinds of data with current state-of-the-art models has not yet been done.

1.2 Problem Statement

Few studies have used ML to forecast the perceived performance of professional female soccer players, with even fewer using both wellness and positional data in a multivariate time series approach. The main research question this thesis seeks to answer is:

> *'What is a good approach to forecast readiness to train among professional female soccer players that result in actionable data for players and teams?'*

From this research question, we find that the overall objective of this thesis is to investigate and discover what type of time-dependent trends and features in our dataset are most important in defining the readiness of an athlete and feed this data through different model configurations to produce useful data. Further, we intend to compare these ML models to determine important mechanisms useful to forecast time series data of this type. To better give answers to our research question, it is necessary to provide smaller sub-questions that, as a whole, provide an answer to our research question.

We use sparse data with a significant number of missing time stamps, which motivates the idea of using complex imputation to see if such an approach can contextualize the missing values. We, therefore, ask the question:

Q1 Is the use of complex data imputation useful to our forecasts?

The data configurations input and output windows have a big effect on results. The input window size determines how many prior time steps are used for a prediction, with the optimal size varying depending on the test data and models. The output window size determines how many unknown time steps to predict. Consequently, each increase in output window size makes the predictions more uncertain. To understand the impact of these data configurations, we ask the question:

Q2 What number of prior time steps is optimal to use when making a forecast, and for how many time steps in the future is it feasible to make forecasts?

We want to investigate whether incorporating more features will positively affect our predictions. Therefore, we ask the question:

Q3 To what extent do multivariate forecasts produce better results than univariate forecasts?

Most previous attempts have only used a univariate approach to forecast readiness. When multivariate data was utilized, the data was limited to a single data type, such as GPS or subjective wellness data. Therefore, exploring how multiple features with different data types impact readiness is important to investigate, and we ask the question:

Q4 What type of time series features are important when forecasting readiness to train?

A prediction in itself is useless if it can not be utilized in a practical application. Therefore, we are exploring several ways of predicting readiness to provide statistics that can aid in decision-making. Our question is:

Q5 What is the viability of our selected use-cases, and do they have the potential to provide actionable data for coaches and players to use?

1.3 Scope and Limitations

The scope of this thesis is to use a combination of features describing the overall physical and mental state of Norwegian female soccer players to forecast readiness to train, a subjective metric describing how well players perceive their abilities to exert themselves physically. We provide a comparative study showing the difference in performance for each of our models on several configurations and investigate different data configurations and use-cases to find practical approaches to forecast readiness. However, this approach is limited to the sport soccer and data/features gathered by the PmSys system [31].

Our dataset is relatively small and consists of time series such as subjective wellness data and objective features derived from GPS data. The data was collected over a period of two years from two different teams. The dataset is also affected by missing time steps (days), resulting in less consistent data.

Much of our data is derived from a self-reporting system called PmSys [31] that collects wellness data from each player. Apart from features like the number of hours slept, the wellness data is subjective and based on the given athlete's quantification of how they feel. The error in subjective wellness reports is the difference between the given answer and the true answer, which is unknown. Therefore, the usefulness of our predictions for real-world use-cases is determined by the error being small enough and the data being a good enough representation of the actual condition of the players. Further, GPS data, which we discuss in Chapter 2.2.4, are prone to inaccurate readings resulting in the data being an approximation of the players' movements.

For our experiments, we have chosen a selection of models that we have deemed promising for time series forecasting. However, this is only a selection, meaning other architectures or models might prove more accurate. Therefore, terms describing the best or most efficient model are limited to our selection of models. Further, we discuss our selection of models in Section 2.5.

In the current state of the available data, the computational cost for running even the most complex models among our selection is not a huge concern. However, if more years of data and features were to be added in the future, then computation could prove an issue.

1.4 Research Methods

The research methods applied in this thesis were predominately quantitative and experimental in nature. The generated results from our experiments were quantitatively analyzed and further evaluated with statistical methods. Our experiments and pipeline are a product of experimental and iterative prototyping [7, 8, 10]. The goal and implementation of each experiment are explained. Each experiment is either an investigation or a hypothesis that is either validated or not. The pipeline was iteratively improved by observing the results from our experiments.

We chose these research methods as we did not know the outcome of our experiments and needed to make improvements based on our findings. Quantitative research allows for comparisons using statistical methods to determine factors like optimal data and model configurations. Further, experimental prototyping ensures clear and testable experiments, while iterative prototyping supports the sequential refinement of our prototype.

1.5 Ethical Considerations

This thesis centers around using wellness and positional data from professional Norwegian female soccer teams [39]. The use of personal information follows a responsibility to safeguard the participants' data and give them agency by properly informing them of the data's intended use [45]. Therefore, the data is made anonymous. To be more specific, this means the removal of all metadata and the use of randomly generated file names. Each athlete has also given their consent and knows what data is collected and in what way it is being used. Further, this data has been exempt from further demand of consent from users since it is anonymous and certified by the Norwegian Privacy Data Protection Authority.

1.6 Main Contributions

The work in this thesis is meant to establish fundamental knowledge important to forecast readiness to train among professional female soccer players. Gaining insight into the future perceived performance of athletes will help players improve athletic abilities, avoid injuries, and aid in-game and training strategy decisions. Some attempts exist to predict readiness for professional female soccer players but lack a comprehensive study regarding complex imputation and the use of multivariate data [34, 46, 60]. We propose a pipeline to use and process wellness and GPS data to generate experiments meant to give insight into optimal data configurations and model selection. We also present several use-cases and a comprehensive statistical analysis of our dataset.

- We present a pipeline to run experiments related to forecasting readiness to train for professional female soccer players. The pipeline consists of a data extraction step (extracting relevant tables from a database), a data analysis and pre-processing step (data is analyzed, imputed, and cleaned), an experiment step (use different data and model configurations to generate specific results), and a results step (visualizing and storing results).

- We provide an extensive statistical analysis of the data, describing the time series qualities of the data, relationships between features, data distributions, effect on data using imputation, and feature importance.

- We leverage complex imputation to determine its effect on our forecasts compared to not using imputation. Through these comparisons, we discover a unique method using imputed data that improves our results.

- Through our experiments, we present three use-cases: Predicting readiness between values one and ten, classifying peaks, and classifying positive, negative, and neutral change in readiness.
- The generated experiments are evaluated quantitatively using a variety of performance metrics. To get a better sense of the impact of our methods, we look at several baseline results to see how our approach compares to trivial methods.
- The best performing models for regression experiments LSTM, XGBoost, and Linear Regression achieved consistently the same error rate. However, for classification tasks, the model with the highest accuracy and F1-score varied for each use-case.

The novel methods presented in this thesis include time series forecasting using multivariate data consisting of different data types such as subjective wellness and GPS data, a unique forecasting method using imputed data that achieves better results compared to not using imputed data, and a use-case that classifies the positive, negative, and neutral change in readiness. Our approach provides extensive insight into key areas important to forecast readiness.

1.7 Thesis Outline

In this section, we will briefly describe the main topics of all chapters found in this thesis.

Chapter 2- Background lays out the necessary theoretical knowledge relevant to our thesis. Here we present the fundamental concepts within the field of ML, different ML algorithm types, and our selection of ML models. Further, we describe athlete health and performance monitoring methods and why they are relevant to this thesis. We present our dataset and how the data was collected. Lastly, previous work, such as time series analysis in soccer, is explored.

Chapter 3- Methods presents how we intend to implement our pipeline. The pipeline comprises four steps: data importing, data processing, experiments, and interpretation of results. We give a description of the system specifications along with describing the evaluation metrics.

Chapter 4- Analysis and Data Processing chapter describes the statistical properties of the dataset used in this thesis and how we alter the data with data processing methods.

Chapter 5- Experiments- First Iteration presents the results acquired by conducting the experiments explained in chapter 3. We explain the significance of each experiment result before discussing what we learned from implementing our pipeline and how we can make possible improvements to it.

Chapter 6- Experiments- Second Iteration improves on what we learned from implementing the first iteration of our pipeline. We conduct the experiments we think are still relevant and discuss their significance. We compare both approaches to further understand what methods have the most positive outcome for these types of time series forecasts.

Chapter 7- Discussion further elaborates on the results gathered from Chapter 5 and 6 to give insight into their significance.

Chapter 8- Conclusion summarizes this thesis and proposes future work based on our findings.

Chapter 2

Background and Related Work

Incorporating strong defining features of an athlete's performance in a multivariate time series ML approach to make automated analyses may save time and resources and lead to beneficial training and game strategy changes. This chapter describes the underlying theory to create such a system in the case of soccer. First, we will go over relevant terminology and explain concepts in athlete health and performance monitoring. Further, we will examine some rudimentary ML concepts relevant to our problem. We will explore several approaches to ML architectures designed for multivariate time series data. We will also discuss the data collection process and the self-reporting system PmSys from which our ML model will be attached and fed data to perform its predictions. Next, we will look at what has previously been done in the field of time series analysis of soccer players using an ML approach with subjective wellness and positional- data. Lastly, we will end the chapter with a summary.

2.1 Terminology

Words have different meanings depending on the context. Therefore, it is important to find common ground and understand the definitions of these key concepts in our thesis:

- **Session** is defined as either a soccer match or training session. This is a continuous period of time that normally lasts up to two hours.
- **Active period** refers to the date period (in days) when players started answering the survey that generates wellness data until they stopped answering the survey.
- **Imputation** is a technique used to replace missing values with an alternative substitute in order for the dataset to retain as much information as possible. There are various methods for imputation, and their complexity can vary depending on the technique used. For example, a simple method can be to replace missing values with the average value. A more complex method is to use ML to predict the missing value based on previous data.

- **Soccer club** or simply association, is defined as an organization consisting of players, coaches, and other parties retaining to a specific soccer team.
- **Forecast** is the word used to refer to a prediction in the time series domain. Forecast and prediction are used interchangeably.
- **readiness to train**, also referred to as readiness, is the target value that the work in this thesis aims to forecast. Readiness is defined as a subjective metric of how ready a player feels to take part in a session. Players report readiness on a scale of 1-10, 10 representing the highest perceived readiness possible.
- **Input window** is the number of previous time steps used to make a single prediction. An input window of three represents three days of athlete data in the case of our dataset.
- **Output window**, also called forecast horizon, is the number of time steps past the input data to predict. An output window of three means predicting three days in the future relative to the latest sample in the input data.
- **Peaks** in this thesis is referred to as very high or very low values of readiness. Peaks are mentioned in related work and the work done in this thesis. Our definition of peaks is readiness values four and below and readiness values eight and higher.
- **Actual days** refers to the data samples only based on days the players answered the survey and provided data. This definition does not extend to days that are a result of imputation.

2.2 Athlete Health and Performance Monitoring

In sports, especially soccer, tools to monitor athletes' physical and mental state are used to obtain information to enhance player performance and avoid the risk of injuries [1, 53, 60, 61]. In this section, we will explain different types of performance monitoring and why they are important to gain insight into the well-being of athletes.

2.2.1 Wellness Reporting

Wellness reporting allows for the internal conditions of the athlete to be monitored by having them give a score of their perceived wellness across multiple areas, such as muscle soreness, stress, fatigue, and sleep. Together these values are often used to calculate a total wellness score [61].

As stated by Wing et al. [61]: The use of wellness reporting is widely regarded as useful in sports science. Many find wellness reporting a better metric than objective ones, such as heart rate measures and blood markers [61]. However, the overall usefulness of the wellness reporting depends on the specific questions providing meaningful data relevant to the given sport [53]. The wellness questions should also be general enough to give an understanding across the entire team and limit the time needed to complete them. Reducing the number of questions and restricting responses to numerical scores is necessary to optimize the time used on surveys. [53].

2.2.2 Training Load

Load monitoring tools help identify how an athlete adapts to a training program and give insight into the risk of fatigue or injury to minimize them. To fully understand and effectively use training load metrics is a complicated process as very few of these metrics have substantial scientific evidence and because no definitive metric exists [22]. Rather, it is important to derive the monitoring systems from what is most relevant to the given sport, and that gives meaningful data about the individual [22].

2.2.3 Injury

Injury and Illness frequently occur to players in varying degrees and will inevitably impact an athlete's performance and well-being. To combat this issue, finding patterns and trends describing possible causes might prevent unwanted developments. FIFA has long conducted injury prevention programs targeting areas that often cause problems [1]. They found that using these programs reduced the overall risk of any injury by 34%. For specific exercises such as the Nordic Hamstring exercise (NH), the long-term result was 51% less hamstring-related injuries compared to teams with no injury prevention programs [1]. This suggests that finding patterns leading up to injuries and making preventative programs can substantially reduce the risk of injury.

2.2.4 Positional Data

In soccer, players are running on a large field trying to score a ball in the opposing team's net. It is the movements of the players that eventually dictate the outcome of the game. Understanding these movements makes it possible to gain insight into the physical state of athletes as well as game strategies. The positional data can be transformed into sports metrics such as top speed and total distance traveled [49]. These are objective measures of training load indicating the intensity of the given session. Using positional data can lead to more informed decision-making and improve player and team performance on the field.

A Case Study on Acquiring Positional Data

Positional data can give important insight into an athlete's performance [44, 49], but challenges arise when acquiring accurate locomotor movements, as stated by Pettersen et al. [44]. The paper compares GPS to Local Position Measurement (LPM) that generates positional data from professional soccer players. While GPS only receives signals, LPM emits them to local receivers that do the necessary triangulation. GPS has been the preferred method in wearable positional technology by clubs, but as evident in Pettersen's paper [44], other technologies like LPM might be a better alternative. The use of LPM over GPS has shown better accuracy, and factors like clouds and the satellite's angle to the ground also do not affect data gathered using LPM.

Multiple studies were conducted to test the accuracy of both GPS and LPM. In the first study, six high-level female players wore two GPS and two LPM tags and performed the Copenhagen Soccer Test for Women. The subsequent distance measured was 11,668 ± 1,072 m with CV of 6% and 10,204 ± 103 m with CV of 1% for GPS and LPM, respectively. For high-intensity runs, which means a speed greater than $16kmh^{-1}$, the results for GPS and LPM were 612 ± 433 m with CV of 37.4% and 1238 ± 38 m with CV of 3.1% respectively. From the first study, it can be concluded that GPS has far more spread in its data points which becomes even more apparent when the athlete's speed is increased. Having two tags of each makes it possible to test inter and intra-reliability. The paper found that the difference measured between the two tags on the same player using GPS data ranged between 800 and 2,071 m, while the LPM data was between 25 and 290 m. This suggests that the data from the GPS consistently measure a higher total distance, which is further supported by Johansen et al. [32] where 19 players wore the same tags and obtained an average distance of 10,805 ± 847 m using GPS data compared to 9,891 ± 974 m from using LPM data. Moreover, the second study supports the results found in the first study, where GPS-measured data continues to measure much larger distances for High-intensity runs-related tests.

The accuracy of both devices was also tested by making the players run along the edge of the soccer field wearing all four tags. The athletes' path was drawn on top of the playing field, and the resulting two images differed significantly. Both images have curved corners, which is expected as the players will not run at an exact 90-degree angle. However, the LPM image is much more accurate than its counterpart, which struggles to stay on the edges and keep its lines from curving out. Several forms of interference can play a part in the poor performance of the GPS. For instance, clouds and fog can cause inaccurate data. The tests took place 69.65 degrees north, and the GPS satellite's inclination from orbit is 55 degrees north and south, which means there were no satellites directly overhead. However, the error of identical tags producing different results has occurred for other GPS models at more optimal orbits. The paper concludes that LPM shows superior accuracy over the GPS [44]. Still, it is unsure if the worse accuracy impacts the GPS's usefulness in quantifying the athlete's performance.

2.3 Machine Learning Fundamentals

ML is a field in computer science and part of the umbrella term artificial intelligence (AI), consisting of algorithms that find patterns and try to improve learning by observing data [23]. These algorithms create unique models based on sample data to make predictions or decisions it was not explicitly programmed to do. ML has been present since the mid-20th century and was first coined by Arthur Samuel in the 1950s [52]. ML has seen a resurgence over the past decade with tremendous leaps in accuracy and scalability with advances in hardware and algorithms [21]. Moreover, models are tailor-made to specific areas and data types. Architectures like convolutional neural networks are often used for computer vision and time series analysis [12], and transformers are making great strides in natural language processing [58]. Next, we will delve into several concepts in ML to establish the required knowledge to understand this chapter's more intricate parts.

The general approach to perform ML is to collect data, carry out data cleaning if needed, train the model, test it on unseen data, and review its performance. In all these steps, there are a number of factors and methods to consider, detailed further in this section.

2.3.1 Domain Awareness

To properly use ML, it is necessary to focus on the models, the data, and the domain in which one seeks answers [30]. Since the idea behind ML is to learn algorithms to solve problems, the problems the algorithms solve must be as close to the real-world problem as possible. This means having good data sufficiently representing the domain.

2.3.2 Data Collection

The usefulness of the model output is highly dependent on the quality of the input data [30]. Extracting data in general but also from several sources might lead to a dataset containing missing data or data with different formats, making data cleaning necessary. One also needs to consider how the data was acquired. If the data is a poor representation of the domain, then the subsequent prediction will have little practical value. Having more data benefits models. However, it is important to consider that more data also leads to a higher computational cost which might be essential to avoid depending on the use-case.

2.3.3 Data Splits

To ensure that tuned ML models can generalize on unseen data it is necessary to split data into three distinct purposes.

- **Train split** is the data used to train the model and should have enough data to generalize new unseen data properly.

- **Validation split** is the data used to check how well the model can generalize unseen data while tuning model parameters to achieve better results.
- **Test split** is the data used to ensure that the model with tuned hyperparameters can still generalize on new unseen data. We want to do this to avoid overfitting.

The validation and test splits must not know anything about the training data to avoid memory leakages, as this would lead to misleading results.

2.3.4 Supervised Learning

We use labeled data to train algorithms to classify data or predict specific outcomes in supervised learning [59]. For example, with an input space X and an output space Y, supervised learning tries to approximate a function f that attempts to map X→Y. Thus, the learning process occurs during mapping X to Y when weights are adjusted until the model is appropriately fitted.

2.3.5 Unsupervised Learning

Unsupervised learning has no overarching programming telling it exactly how it should interpret observations and whether its reasoning is correct. Instead, it will traverse its data and find patterns often by clustering data points the algorithm deems similar enough [59].

2.3.6 Overfitting

Overfitting is a term used when a statistical model fits too well to the training data [59]. The issue is then that the model will not perform nearly as well on other unseen datasets as its ability to generalize is gone. An example would be: Given several data points, the following function of the overfitted model to the training data would be exact lines separating two data classes as shown in the rightmost image in Figure 2.1. However, if the model is not fitted well enough to the training data, we end up with underfitting. Figure 2.1 shows an example of what that looks like. The wanted result is a balanced fit to the data where the model can generalize on unseen data but also fits well to the validation data. An example of this is also visible in Figure 2.1.

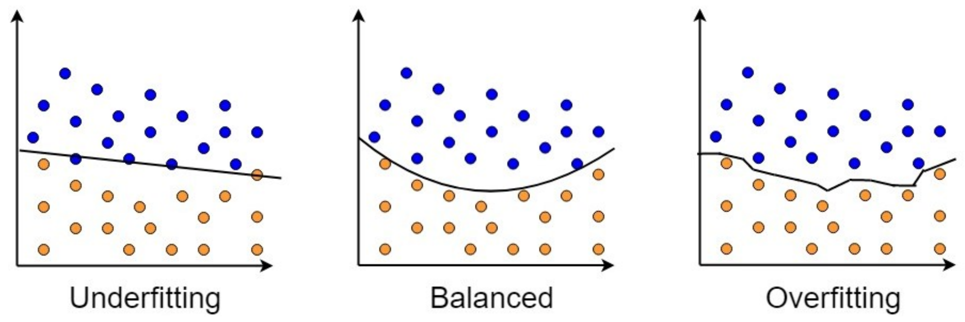


Figure 2.1: An illustration showing three images of a model underfitting the data, overfitting the data, and having a balanced fit to the data.

Interpretable AI

The concept of Interpretability in ML models has become more prudent with the rise of more complex models being a part of more and more people's lives [47]. Explaining how a prediction was made is crucial, especially when sensitive or even personal data is used or when a prediction directly impacts individuals [47].

2.4 Machine Learning - Types of Algorithms

ML is the process of learning algorithms to solve problems. These algorithms vary greatly in both use-cases and complexity. This section will go through different types of ML algorithms relevant to this thesis.

2.4.1 Neural Networks

Neural networks are a subset of ML and an artificial approach to imitating how biological neurons found, for example, in the human brain, send signals to each other. These networks consist of an input and output layer with an N number of hidden layers in between, where each artificial neuron is a node connected to other nodes in the network. In Figure 2.2, we can see an example of what that looks like.

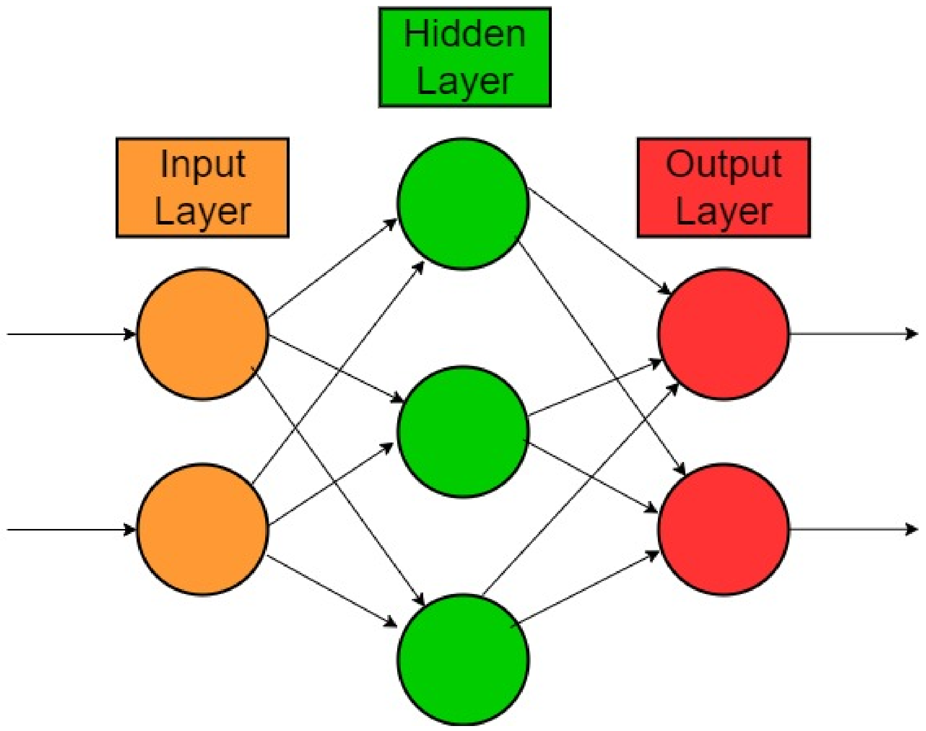


Figure 2.2: An Example of a simple neural network with one hidden layer.

The first trainable artificial neural network was the Multilayer Perceptron (MLP) introduced by Rosenblatt [48] in the 1950s. The MLP was designed as a linear classifier with only one layer between the input and output layers with a threshold activation function. This design differs from modern neural networks, which use non-linear activation functions [19] that allow learning from a wider variety of data, creating more intricate decision boundaries.

The learning process

A neural network has mainly three steps defining its learning process [6]: Forward propagation, error computation, and backpropagation. These processes, in tandem, will iteratively learn and improve the network's overall performance.

Forward propagation

Forward propagation is the process of input data propagating through all the network layers resulting in output in the form of a prediction. The process is initialized by nodes being fed data from all the nodes of the previous layer, which are then combined with a node's own set of weights. Furthermore, for each node, an activation function is applied to calculate what is known as the hidden activation, which will serve as input to the next layer of nodes.

Error computation

When forward propagation has produced a prediction it is necessary to compare this value to the actual prediction the model was supposed to generate. This is known as calculating the error and can be done by using a loss function like squared error $\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^*)^2$. The loss function is applied to adjust the network's weights to better approximate the mapping of X$\rightarrow$Y by using backpropagation.

Backpropagation

So far, we have described the process of generating predictions from input data through forward propagation and finding the error between the predicted value and the true value, known as the loss. Further, for the network to learn, it needs to minimize this loss by adjusting its weights accordingly with a delta in a direction where loss is minimized. This is called backpropagation.

2.4.2 Linear Regression

Linear regression [55] is a regression model that models linear relationships between a response variable and one or more explanatory variables. The motivation behind the linear model is to examine whether the explanatory variables manage to predict the response variable accurately. By doing this, we estimate unknown parameters, which is the process of fitting the model to the data. As with the learning process of all models, it is crucial that the model is able to generalize. This is evident when the model properly adjusts its predictions when encountering new unseen data. Next, we will go through the general mechanisms of multiple linear regression.

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k + \epsilon \tag{2.1}$$

In Equation 2.4.2, we have the equation for multiple linear regression. y is the response variable with β_0 being a constant, referred to as the intercept. X_k is the explanatory variable, and β_k is the slope coefficient for each explanatory variable. Lastly, ϵ is the model's error term, referred to as the residuals.

2.4.3 Decision Tree Regressor

The decision tree model is initialized by the root node consisting of the complete sample, which is further split into several new nodes. These nodes branching from the root node are called interior nodes and represent all the features, with branches being the decision rules. Further, we have the leaf nodes, and these represent the outcome.

When the tree models are given data, each sample is run through each node in the tree until it reaches a leaf node, producing an outcome. When going through each node, the data sample is compared to several TRUE/FALSE questions determining its next node. The outcome is the average response variable value in the specific leaf node. It is through several iterations that the model is able to predict a reasonable value for a data sample.

2.4.4 Long Short-Term Memory (LSTM) Networks

LSTM [27] builds on the recurrent neural network (RNN) architecture to mitigate the issue of exploding and vanishing gradients as well as improving short-term memory. Exploding and vanishing gradients occur as the gradients are a product of past gradients, and these tend to grow exponentially, which leads to weights being too big or unchanged. This culminated in creating the LSTM that works by using cell-states and gates that allow the network to forget or remember certain information. These mechanisms let the model retain long-term and short-term memory, which is important for large sequences of time series data.

2.4.5 Convolutional Neural Networks

A convolutional neural network is an artificial neural network mainly applied to spatial data with a grid-like structure such as image and time series data [2, 12]. Further, we will describe the convolutional neural network's general learning process before discussing convolution and pooling.

The Learning Process

The iterative learning process is similar to the one described in Section 2.4.1, where the input data is forward-propagated through the network. Some loss is calculated and propagated backward in a direction so that the weights can be adjusted to minimize loss. The convolution and pooling layers set convolutional neural networks apart from the simple neural network we have already described.

Convolution

Convolution is a mathematical operation on two functions that produces a third function giving insight into the relationship between them. In a convolutional layer, we have the same number of learnable filters as we have nodes since each node constitutes a filter learning feature. The convolution can be visualized as a sliding window called a convolving kernel over the input data. The convolution output is called a feature map, while a set of feature maps is called the output volume. The feature maps are further processed in a non-linear activation function; otherwise, all the deep convolutional neural network layers would collapse into a single convolutional layer. Moreover, the convolution layer has hyperparameters like size, padding, stride, and dilation, dictating the spatial dimensions of the output volume.

The values and dimensions of the convolving kernel are based on the input data. In the case of images, it would be a 2d convolving kernel with values for examples corresponding to finding edge features. In time series data, the kernel would be of one dimension and roughly give the moving average of the time series, providing insight into a possible trend.

Pooling

Pooling is a commonly used mechanism to reduce the spatial size of the feature maps to allow for more efficient computation. The pooling operation is applied to the feature maps after they have gone through the non-linearity function. The pooling filter is always smaller than the feature maps and results in a size reduction. There are two techniques: max pooling and average pooling. In max pooling, the greatest value in a segment represents that segment, while in average pooling, the average value of all values in a segment is used.

2.4.6 Transformer

The transformer architecture [58] uses an encoder-decoder structure to make predictions. However, it does so without the use of convolution or recurrence. Still, models built on the foundation of the transformer have achieved state-of-the-art results in a wide variety of fields such as Natural Language Processing (NLP), music generation, computer vision, and time series analysis [28, 58, 62, 63]. In the case of NLP, the transformer learns the relationships found in sequential data. It does this by regarding some words as more important than others and therefore is able to learn sentence structure. Further, we will describe the transformer architecture.

The Transformer Architecture

Each input sequence fed through the transformer is first transformed into an embedding vector equal to the model dimension. The embeddings are then augmented with a positional embedding by generating sine and cosine functions that are simply summed to the embeddings. As there is no recurrence, the model depends on positional embeddings to understand the relative positions of the sequences. Further, the embedding vectors are sent through an encoder block containing two layers: The first layer is a multi-head attention mechanism that outputs normalized keys, values, and queries and is fed into a feed-forward network consisting of two linear transformations. The input to the decoder, which is a predicted output at timestep t-1, is also augmented with positional embeddings. The decoder block consists of 3 layers, where the first layer applies masking to the succeeding elements in the sequence. The second layer introduces the output from the encoder block and unmasks all the elements in the sequence. As for the third layer, the output goes through a fully connected feed-forward network, followed by a soft-max layer that predicts the next element in the output sequence.

2.5 Machine Learning - Selected Algorithms

A none-deep ML approach like decision trees [11] offers fast learning but often results in a lower accuracy when compared to a more computationally complex model with multiple layers like convolutional neural networks [18]. Further, by understanding how different models learn, it might be possible to extract information on why some mechanisms are better suited to athlete time series data than others. This section aims to understand the models when we later interpret the results in chapters 5 and 6. Having insight into the details of each model and why it performed the way it did might give a better understanding of what to change to allow for better results. Therefore, we intend to use ML models with different architectures and complexity that have shown state-of-the-art results using time series data. We will also use less complex models like the decision tree and linear models as a baseline for comparisons. Next, we will present our selection of models that we will be using for our comparative study.

2.5.1 eXtreme Gradient Boosting

eXtreme Gradient Boosting or XGBoost is a tree-based algorithm highly used by data scientists for various ML tasks [11]. The model saw its rise in adaptation through ML competitions where it was widely used and achieved state-of-the-art results [11]. Next, we will go through the model architecture and explain its unique mechanics, which makes it perform well with different kinds of data.

XGBoost is based on the boosting machine, a sub-group of Tree-based Ensemble algorithms similar to random forest. However, random forest uses a parallel approach to learning by creating several decision trees and using the average of the aggregated result from each tree. XGBoost, on the other hand, employs a sequential learning algorithm by adding one new decision tree at a time. It does this by initializing the first model and then calculating the error of this model; it then creates a new model to predict the error of the previous model, which is then added to the ensemble. At this point, XGBoost will repeat the process of generating new models to predict the previous model error and add it to the ensemble. Despite generating trees sequentially, XGBoost, from a system optimization perspective, is designed for parallel computation to reduce the computational cost and is, therefore, very competitive in terms of performance.

The "eXtreme" part of its name comes from implementing several algorithms that enable XGBoost to fully exploit the available computational components of the given system, such as CPU, GPU, memory, and disc.

Some of these algorithms are out-of-core computation dividing data into blocks to be stored on disc, cache-aware pre-fetching algorithm improving the performance of a factor of two for large datasets (>10 million instances), and block sharding allowing for efficient pre-fetching of data when multiple discs are available [11]. All these methods allow the model to scale with hundreds of millions of examples and to be run on a single desktop.

Aside from creating an end-to-end tree boosting system, There are three significant innovations made in creating the XGBoost model.

- **Weighted Quantile Sketch** proposes possible split points in the approximate algorithm. The idea is to provide a data structure supporting merge and prune operations where each operation can maintain a certain accuracy level.
- **Sparsity Aware Algorithm** considers sparsity in data, meaning missing data, frequent zero entries, and consequences of feature engineering such as one-hot encoding. Therefore the algorithm instills a direction in each tree, making the algorithm aware of the sparsity patterns in the data.
- **Cache-Aware Block Structure For Out-Of-Core Tree Learning:** is a series of algorithms enabling effective use of the memory hierarchy of computers from the cache to disc to more efficiently fetch data.

XGBoost balances variance and bias in the way of L1 and L2 regularization to make the model more proficient in model generalization. Especially compared to gradient boosting, which only considers the variance and is prone to overfitting. Further, XGBoost allows for tweaking a wide variety of hyperparameters such as learning rate, depth of tree, number of trees, and the number of nodes, for mentioning a few. These factors allow XGBoost to improve the variance-bias issue further.

Tree-based models like XGBoost are known to perform exceptionally well on tabular data. When the features are heterogeneous, e.g., number of sales, weight, and average time spent. For these data types, XGBoost or other tree models like random forest tends to outperform neural networks [11]. This can be explained by how XGBoost handles features independently, where the rules for generating leaf nodes are expressed as if statements. To obtain a prediction of whether a person is going to buy an item, a tree-based model generates statements such as: "Is the person older than 30" and "Is the person married". A neural network on the other hand could, for example, model the probability of a sale such as $Sale = W_1 * age + W_2 * married$. Generally, finding meaning by linearly combining features very different from each other is a difficult task and explains why tree-based models are still very relevant today.

2.5.2 ROCKET

ROCKET [12] uses random convolutional kernels to transform time series data into features to train a linear classifier. Most models achieving state-of-the-art accuracy on time series data are computationally expensive. Still, examples of better scaling deep architecture like InceptionTime [29] and time series Transformer [63] are prevalent. However, as the paper points out, ROCKET uses only a fraction of the computation of such models, as is evident when comparing test results from the 85 "bake off" datasets from the UCR archive. While InceptionTime spent more than six days to complete the task, ROCKET only needed 1 hour and 50 minutes while also achieving state-of-the-art results.

ROCKET is made scalable and programmed to run in parallel on CPU cores or GPUs automatically. The architecture, with its random convolutional kernels and linear classifier, is, in reality, a 1-layer convolutional neural network. ROCKET applies either a logistic or ridge regression depending on the dataset size: Primarily using ridge regression and only logistic regression when the number of training examples is far greater than the number of features, as logistic regression trained to utilize stochastic gradient descent scales better on larger datasets.

The paper notes four main differences in the convolutional layer common in typical convolutional neural networks: Firstly, there is only one layer of kernels, and their weights are not learned, so the cost of computing convolutions is exceedingly low. Therefore ROCKET uses an enormous amount of these kernels. Secondly, all aspects of the kernels are random. This means that length, dilation, padding, bias, and weight are all given random values. It is typical in convolutional neural networks that dilation increases exponentially with depth. However, dilations in ROCKET are randomly sampled from each kernel, giving a large variety of dilations that capture patterns at different scales and frequencies. Lastly, ROCKET uses what the authors call "the proportion of positive values" to let a classifier weigh a pattern's commonness within a time series. This is described as the most important aspect of the architecture concerning accuracy. Hyperparameter K defines the number of kernels that directly impact the computation cost and performance of the model. A lower K results in lower accuracy but less time for training, while a higher K has the opposite effect with higher accuracy but requires more time to train.

In the TST paper [63], ROCKET was compared to other state-of-the-art MTS models on multivariate time series datasets from the UEA archive and performed best in 3 out of 11 datasets and overall was the second-best model in terms of rank. ROCKET proves that using a large number of random kernels to discriminate patterns in time series data is extremely efficient in terms of accuracy and computation.

2.5.3 Temporal Fusion Transformer

The Temporal Fusion Transformer (TFT) [35] is a transformer-based model created for interpretability, high performance, and multi-horizon forecasting. It leverages self-attention mechanisms to map complex temporal dynamics across multiple time series.

TFT is unique in that it supports different kinds of features consisting of time-varying known and unknown as well as time-invariant real and categorical. Holidays are an example of a known time-varying variable, while we categorize something like CPI as an unknown time-varying variable. Further, ID is a categorical time-invariant variable, while the yearly inflation rate is a real-time-invariant variable. The purpose of accommodating these feature types is for the model to learn their temporal relationships. Further, what sets TFT apart from other models is that it offers interpretability for these forecasts using different variable types. Previous work was plagued by "black-box" solutions [35]. These novel properties are a product of the model's unique architecture.

The most contributing parts of the TFT architecture [35] are the following:

- **Gating mechanisms** disregard unused parts of the architecture to allow for adaptive model complexity. This mechanism ensures that the model is able to take into consideration extensive examples of scenarios and datasets.
- **Variable selection networks** chooses relevant input features at each time step.
- **Static covariate encoders** incorporates static features into the network.
- **Temporal processing** is used to learn long and short-term temporal relationships between known and observed time-varying inputs.
- **Prediction intervals** leverages quantile forecasts to approximate a range of probable target values at each prediction horizon.

The main contribution of the TFT in the time series domain is how it offers an interpretable and versatile model for multi-horizon forecasting. Its architecture supports interpretability regarding global feature importance, temporal patterns, and significant events. By temporal patterns, we mean a segment that recurs frequently, for example, a combination of chords in a musical piece. By learning these patterns, the model can also detect abrupt changes in temporal patterns that might signal a significant event. An added focus on the temporal relationship between variables makes the TFT model suited for complex time series forecasting.

2.5.4 Additional Models

In addition to the models presented in this section, we will use linear models, a simple decision tree model, and an LSTM model for our experiments. These have previously been presented and discussed in section 2.4.

2.6 SoccerMon Dataset

In this section, we present the soccermon dataset [39] and the framework PmSys [31], where we obtain data for our dataset. We will go through how the data was collected and explain the subjective and objective nature of the data.

2.6.1 SoccerMon Subjective Metrics

Subjective wellness feedback is an essential and much-used method to track an athlete's performance and wellness. Knowing how well an athlete performed and overall felt during previous training sessions and competitions makes it easier to keep track of progress and how best to make improvements for further athletic development and prevent injuries. The subjective wellness data can be multiple numerical values to an athlete's sleep, stress, and soreness or comprise all these values into a single score. As for this thesis, our dataset regarding subjective wellness data consists of numeric values to perceived wellness across multiple areas as shown in Table 2.1. Table 2.2 shows subjective training load metrics.

Metric	Description	Data Type	Range
Fatigue	The current fatigue level of the player	Numeric	1-5
Mood	The current mood of the player	Numeric	1-5
Readiness to play	The athlete's readiness for a training session or a game	Numeric	1-10
Sleep duration	The duration of the sleep	Numeric	0-12
Sleep quality	The quality of the sleep	Numeric	1-5
Soreness	The level of soreness	Numeric	1-5
Stress	The current stress level of the player	Numeric	1-5

Table 2.1: Wellness metrics in the SoccerMon dataset.

Metric	**Description**	**Formula**
Session RPE (sRPE)	The workload of a single session depending on the duration and the reported RPE values	$RPE \cdot duration$
Training load	The sum of sRPE during a day	$\sum sRPE$ per day
Weekly Load (WL)	The sum of sRPE over the last 7 days	$\sum sRPE$ per week
Acute Training Load (ATL)	The current level of fatigue (average sRPE over the last 7 days)	$7^{-1} \sum_{n=i}^{i+7} DL_i$
Chronic Training Load (CTL)	The cumulative training dose that builds up over a longer period of time (average sRPE over the last 28 or 42 days)	$x^{-1} \sum_{n=i}^{i+x} DL_i$, x = 28 or 42
Acute Chronic Work Load (ACWR)	An indication of whether an athlete is in a well-prepared state, or at an increased risk of getting injured (ATL divided by CTL)	$ATL \cdot CTL^{-1}$
Monotony	Reflection of training variation across the last 7 days (mean sRPE divided by the standard deviation (SD) = ATL / SD)	$ATL \cdot SD^{-1}$
Strain	Reflection of the overall training stress from the last 7 days (total weekly sRPE multiplied with Monotony)	$WL \cdot Monotony$

Table 2.2: Training load metrics in the SoccerMon dataset. For a training session, the players report session duration and the overall Rating of Perceived Exertion (RPE). These subjective parameters are then used to calculate a variety of different measures of training load as shown in this table.

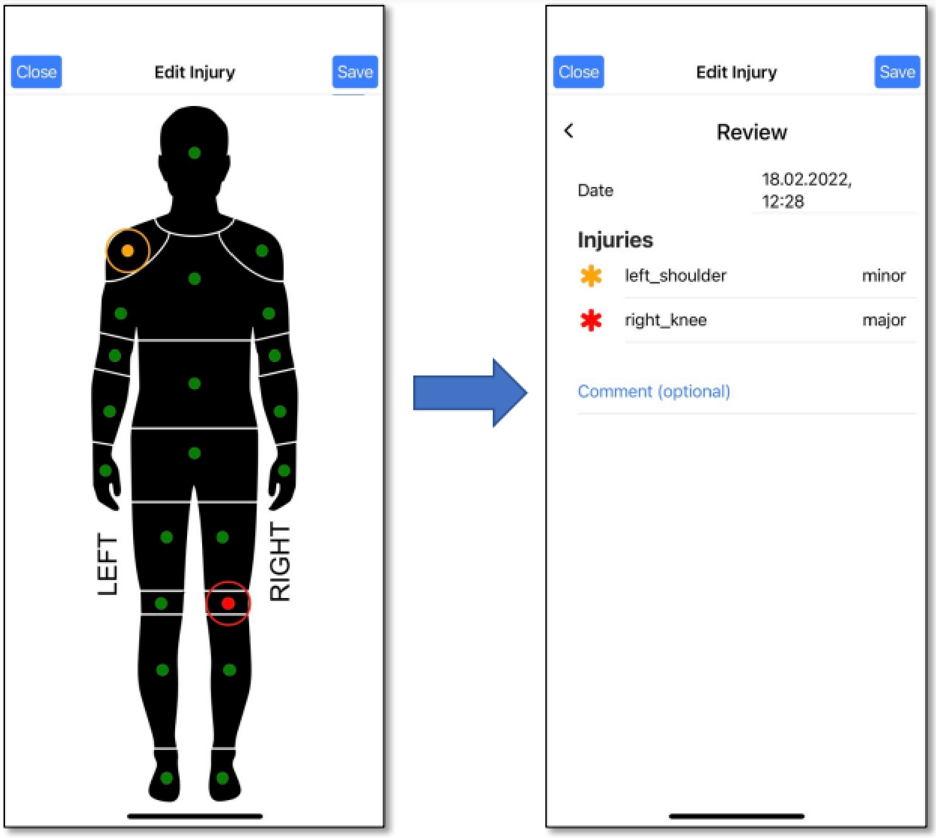
2.6.2 SoccerMon Objective Metrics

Positional data obtained through, for example, GPS gives data on movement by measuring athletes' position several times a second. Such data can, in turn, be used to calculate total distance, the number of accelerations above a certain speed, and average speed, to mention a few. These types of data are objective and show the true performance of an athlete. However, the importance of understanding athletic development might vary immensely between players. This is why it is crucial to make systems that consider each athlete's subjective nature.

2.6.3 PmSys Framework

PmSys [31] is a smartphone-based monitoring and reporting system athletes use to gather and analyze subjective wellness data to gain insight into an athlete's development and avoid injuries. Unfortunately, creating and supporting a smartphone application is costly and resource-demanding. Still, the Johannes et al. [31] argue that focusing on user experience to reduce time spent reporting each day is essential for users to want to use the application. This statement is also supported by other literature [53].

Several improvements and changes were made to avoid tedious reporting, such as making a point-and-click body silhouette to report injury rather than answering 11 complex questions. An example of this can also be found in Figure 2.3. Text and button clicks were reduced, questions simplified, and scrolling eliminated. These were among the changes that were deemed essential to ensure a better user experience. In Figures 2.4 and 2.5, we see two images showing how each athlete reports wellness and training load utilizing these improved functionalities for faster reporting.



Date 18.02.2022, 12:28

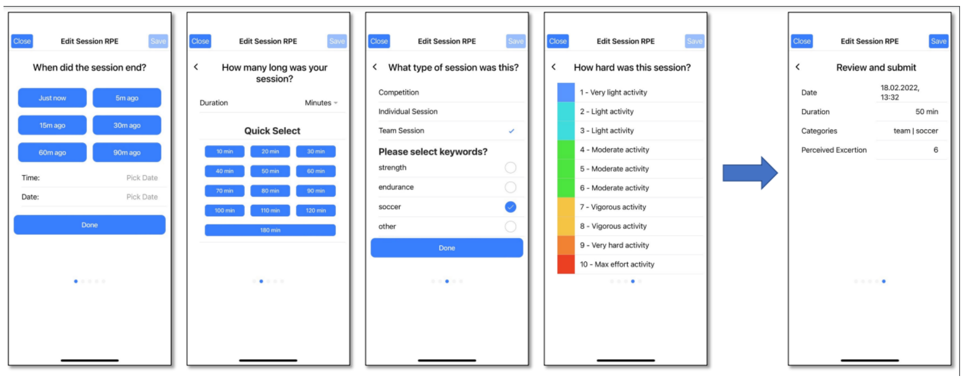
Injuries

✱	left_shoulder	minor
✱	right_knee	major

Comment (optional)

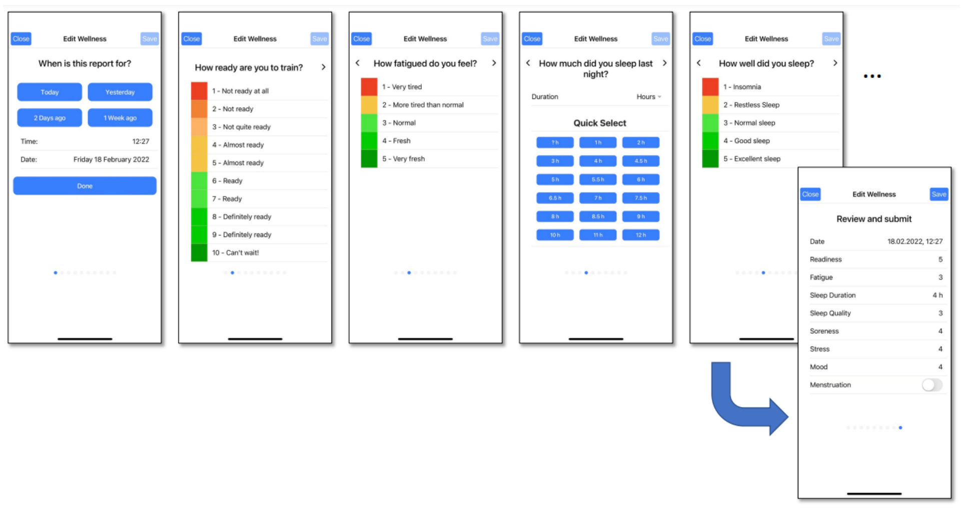
Figure 2.3: An image from the PMSys app illustrating the point and click body silhouette for reporting injuries. The user indicates what body part is affected and the severity of the injury (courtesy of ForzaSys).

PmSys collects data from the athletes' answers to questions regarding their training load, wellness, and mood. These data are further analyzed and are available for coaches and physicians through a trainer portal. The coaches can view individual players' and teams' statistics through the trainer portal and send direct messages to the player's PmSys profile [44]. A key component of PmSys is the dynamic between the coaches, analyzed data, and athletes that the system support to give better insight and educate all parties. The system's intended use is daily, and the time to report is right after waking up to the start of the first training session.



Date	18.02.2022, 13:32
Duration	50 min
Categories	team | soccer
Perceived Excertion	6

Figure 2.4: An illustration of how training load is reported by users in the PMSys application (courtesy of ForzaSys).



Date	18.02.2022, 12:27
Readiness	5
Fatigue	3
Sleep Duration	4 h
Sleep Quality	3
Soreness	4
Stress	4
Mood	4
Menstruation	

Figure 2.5: An illustration of how wellness is reported by users in the PMSys application (courtesy of ForzaSys).

PmSys is being actively used by junior, elite, and national teams in Norway. Additionally, 400 female soccer players from Norway, Denmark, and Portugal are part of a research project on the development of elite female performance using the PmSys framework. The nature of the collected data makes it compatible with time series analysis because it is measured over a consistent time interval. Good data is crucial for a ML model to learn and output reliable predictions. The paper found promising initial results when predicting future athlete performance based on the data from PmSys. More details on PmSys time series data predictions are described later in this chapter. Further, the paper states the reaction by the athletes and coaches to be positive, and several changes in strategy have been made due to the information given by PMSys [31].

PMSys incorporates intelligent and efficient methods of extracting athletes' training load and wellness data to give insight into their performance. Statistics and other visual representations of their athletic state are used to simplify and abstract time series data measured over a consistent time interval. Adding an automated ML analysis component to predict future athletic performance could draw more information about an athlete's true state. Moreover, utilizing multiple features with different characteristics to profile athletes might provide more context to describe performance better. PMSys is based on self-reporting of mostly subjective wellness data, which is only as accurate as the questions and the user's reporting.

2.7 Time Series Prediction for Soccer

2.7.1 Time Series Data

Time series data is chronological data measured consistently over an even-spaced interval of time and is used to discover trends that may indicate the future movements of the data. These movements are used in a number of fields to gain valuable insight, such as weather forecasting [9] but also in analyzing athletic performance [60], which is the focus of this thesis.

Finding and analyzing patterns in time series data allows for predicting likely outcomes with a significant degree of certainty. In an athlete's performance context, these predictions can help avoid injury and contribute to knowledge used to change and find better matches and training strategies. There are already cases where analytical data in sports have been utilized to alter game strategies [31]. Further in this section, we will present examples of what has been previously done in analyzing time series data with an ML approach in the world of soccer.

2.7.2 Predicting Peak Readiness-to-Train

Using time series data to predict important events is not a new concept. It has already been utilized to predict peak readiness among professional soccer players [60]. Wiik et al. have athletes submit subjective wellness data, which is used to train a LSTM [27] model that predicts positive and negative peaks. The readiness-to-train value serves as input to the LSTM model, where the output is the predicted readiness value for the next day as the model works on a day-by-day basis. The readiness value is between 1 and 10, with positive peaks being everything eight and above and negative peaks three and below. Negative and positive peaks were used as predictions rather than trying to predict a specific number accurately. As the paper states, this was done to mitigate the problem of limited data.

The model was tested on two teams, and each individual was given a predicted readiness value based on their data and the combined data of the team excluding themselves. This led to the result where the predicted values were more accurate on the team's data than on their data. However, they argued this to be because the data available for each individual was too sparse and that the team's dynamic influences the player's performance. Using these methods, they predicted positive and negative peaks with recall and precision above 90%. Serving as proof that it is possible to predict an athlete's future performance with a significant degree of certainty. However, a model that can better predict an athlete's performance based on only their data is still desirable since each player is unique and will respond to training parameters differently.

2.7.3 Predicting readiness to train Using LSTM

This thesis further builds upon previous work regarding predicting athlete time series data which mostly based their research on the LSTM architecture with a univariate approach [34, 46, 60]. Their research shows promising results and concrete ideas to further contribute to this field.

Ragab [46] used the LSTMPlus model from the TSAI library [41] and achieved the best results by training on the team's data and making forecasts for each individual player. These predictions were made for two unique teams: A and B. Team A had a higher player consistency, leading to more accurate forecasting, and team B had lower player consistency. Further, they discovered that utilizing more data and training only on individual players resulted in worse results than using substantially fewer data samples and training on the whole team. Generally, they discovered that more data did not increase accuracy and that less and more recent data is preferable to more data that is not so recent. As the LSTMPlus supports multi-step, forecasting multiple days in the future was attempted, leading to less accurate results with each increase in forecasting horizon. An increase in input size did not mitigate the issue. Regarding the hyperparameters, significant deviations were not found by changing them except for turning shuffling on, which yielded slightly higher accuracy. Further work points to other ML models and variables for multivariate forecasts.

2.7.4 Exploration of Different Time Series Models

Kulakou et al. [34] used several ML models to predict future readiness to train among professional female athletes using both a univariate and a multivariate approach. The paper builds on the findings from Wiik et al. [60] and utilizes the same system (PmSys) to extract training data from two different Norwegian elite female soccer teams over two years. They used ML models from the TSAI library [41].

Kulakou et al. investigated multiple areas important to athlete predictions using time series data: Multiple ML models were employed (RNN, LSTM, GRU, RNNPlus, LSTMPLus, GRU-Plus, RNN-FCNPlus, LSTM-FCNPlus, GRU-FCNPlus, InceptionTime, MRNN-FCNPlus, MLSTM-FCNPlus, MGRU-FCNPlus) with univariate- or multivariate data using readiness only for the univariate approach and mood, stress, soreness, fatigue, and readiness for the multivariate approach. Trained models were given different window sizes on the test data to discover how much input data was needed for predictions and if longer predictions were viable. As was also done in Wiik et al. [60], Kulakou also compared training on individual players and the whole team. Further, they investigate the impact of hyperparameters along with the issue of missing values.

Kulakou found in their experiments that a multivariate approach did not perform better with their chosen models and dataset. However, they encourage future work to explore a more comprehensive approach to multivariate data. The best-performing model using multivariate data was inception time, while LSTMPlus was the top performer using univariate data. A smaller output window size proved to be more accurate, meaning that daily predictions were more accurate than weekly predictions. As Wiik et al. discovered [60], training on the entire team rather than on each player proved more accurate in finding positive and negative peaks. Other than enabling shuffling, changing hyperparameters had no significant impact on the results. Lastly, the missing data were treated with and without gaps, but the most accurate results were found using missing values without gaps.

2.7.5 Injury Forecasting With GPS data

GPS-tracking allows measuring an athlete's speed and distance during games and training sessions. This data can, in turn, be translated into training load features and used to profile athletes for future performance and injury prevention. Alessio's paper [49] proposes methods to effectively utilize GPS data to analyze professional soccer players by extracting 12 features to be used in a multi-dimensional model that forecasts whether or not an athlete will sustain injuries based on the most recent training load. The 12 workload GPS features extracted are categorized into three groups: (1) Kinematic features describe the overall movement of an athlete. (2) Metabolic distance features quantify the energy expenditure of an athlete's movement. (3) muscular-scheletrical load gives a general measure of load on the body. Together with these 12 features are another 43 that consist of personal data such as age, and the number of occurred injuries, Exponential Weighted Moving Average features, Acute Chronic Workload Ratio features, MSWR features describing the monotony of the workload features, and the relationship between a current training session and previous injury.

All features mentioned comprise the dataset and are further processed to find the best features by reducing the feature space dimensionality, limiting the risk of overfitting. This is done by performing a feature selection process called Recursive Feature Elimination with Cross-Validation (RFECV) [20]. The most relevant ones are the subsequent subset of features that give the highest score on the validation set.

The dataset is used on several different models, but the decision tree is the best-performing one, and a recall and precision of 0.8 and 0.5 were achieved. These results show that the model finds injuries 80% of the time and correctly classifies games/training sessions as injuries 50% of the time. This suggests that the model predicts a non-injury as an injury one time out of every two occurrences, which is substantially less accurate than the best result presented by Wiik et al. [60]. However, Wiik et al. predict extreme readiness values, while Rossi et al. predicts injuries that are much rarer. Also, Wiik et al. use the team's data for predictions rather than only the relevant player, which has been shown to help results significantly.

In addition to being tested with data from a complete season, they tested the model on another season but were only given new data for every new training session/game. This was done to see how the model performed with limited data. The paper reports poor performance the first few weeks, but from week six until end of season, the model finds nine injuries out of 14 with a precision of 0.56. With this, Rossi et al. indicate that it only takes a couple of months of data before the model becomes efficient.

2.8 Chapter Summary

In this chapter, we have presented the basic concepts in ML, different types of ML architecture, and specific ML models that will be used for our experiments. Further, we discussed the importance of health and performance monitoring tools to evaluate players' mental and physical state along with SoccerMon, our dataset. Lastly, we presented related work using wellness or GPS data to forecast athletic performance.

Multivariate and univariate time series analysis of athlete data has proven valuable in injury and performance forecasting. However, the examples shown have mainly used subjective wellness or positional- data. This motivates the idea of combining a multitude of features describing an athlete's performance to use in a multivariate time series ML architecture. Using several state-of-the-art ML models might provide better insight into what type of models is a good approach for these kinds of time series data.

Based on related work discussed in this chapter and their comments on future work [34, 46, 49, 60], it is important to conduct a more comprehensive investigation regarding:

- **Data Configurations** such as an optimal number of input windows and how this value might change depending on the forecasting horizon and model type [34].
- **Exploration of Architecture Types** that looks at models outside the TSAI library [34].

- **More Advanced Methods of Imputation** to better represent the daily time series data [34].
- **A Deeper Analysis of Multivariate Data** to investigate what other data types have a strong explainable relationship to readiness [34].
- **Investigating the Transferability of Data** to see whether it is possible to use data from multiple clubs to train and forecast on an arbitrary player [49].

These bullet points represent research that is missing or not thoroughly investigated [34, 46, 49, 60]. Therefore, we will address these questions as they are important to understand the ability to forecast readiness and are also relevant to our research question. Next, we will describe our methodology in chapter 3.

Chapter 3

Methodology

We explained the health and performance benefits of tracking an athlete's performance in chapter 2, including the concept of ML and its application in forecasting future performance based on previous time series data. Further, a description of several deep learning architectures, related work, and the base data which will become our dataset has also been elaborated on in the previous chapter.

The focus of this chapter is the implementation of our pipeline to conduct experiments. We propose a novel approach using subjective wellness and GPS data with the current state-of-the-art time series models to forecast future readiness to train among professional female soccer players. We will first discuss what is considered a good approach to forecast athlete data to produce useful information for players and clubs. We will then explain each step in our proposed pipeline before giving a technical description of the requirements to implement such a system. Lastly, we will present the evaluation metrics used to interpret our results and present a summary for this chapter.

3.1 Characteristics of Useful Forecasts

To better comprehend our main research question, it is necessary to understand what signifies useful characteristics when forecasting readiness. From what we presented in chapter 2, we have derived the following properties as important when forecasting readiness to train.

- **Data impacting game strategy:** The model output should produce data to help make game strategy decisions. By this, we mean anything from selecting players for matches to adjusting training schemes to improve performance and avoid injury.
- **Personalized to the player:** The model output should reflect each player's unique patterns and variability. Having models only capable of forecasting certain types of players would be detrimental to its real-world use-case. The main focus of this thesis is understanding players on an individual level.

- **Robust models:** The models used to make forecasts should be reliable and not too computationally expensive. By reliable, we mean that the model error is not higher than its usefulness. However, what error is too high is difficult to conclude and will be discussed later in chapter 7.

3.2 Proposed Pipeline

The motivation behind representing our work as a pipeline is to show each unique part of our work and how they are connected. A pipeline approach allows us to enforce a structure on how we go from acquiring data to producing results. We previously explained in Section 1.4 how our research methods are based on experimental and iterative prototyping; We generate results and try to improve them based on what we learned by conducting experiments. An ordered approach to our overall methodology aids us to more efficiently and Rigorously conduct experiments. Figure 3.1 shows the pipeline describing our workflow.

To thoroughly investigate what elements are important when forecasting athlete data, we must determine methods for reducing forecasting error and consider what constitutes useful data that clubs and players can act on. We have previously discussed in sections 2.2 and 2.6 the benefits of knowing the future well-being of an athlete to optimize training and potentially avoid injury. However, the format in which model output is expressed is also crucial to any data analyzing tool. Therefore, we will do regression and classification to explore different use-cases and ways of representing our results. We will provide two iterations, with a first approach and a second approach that builds on what was learned in the first iteration. Next, we will go through each step in the pipeline, from acquiring data to visualizing the results.

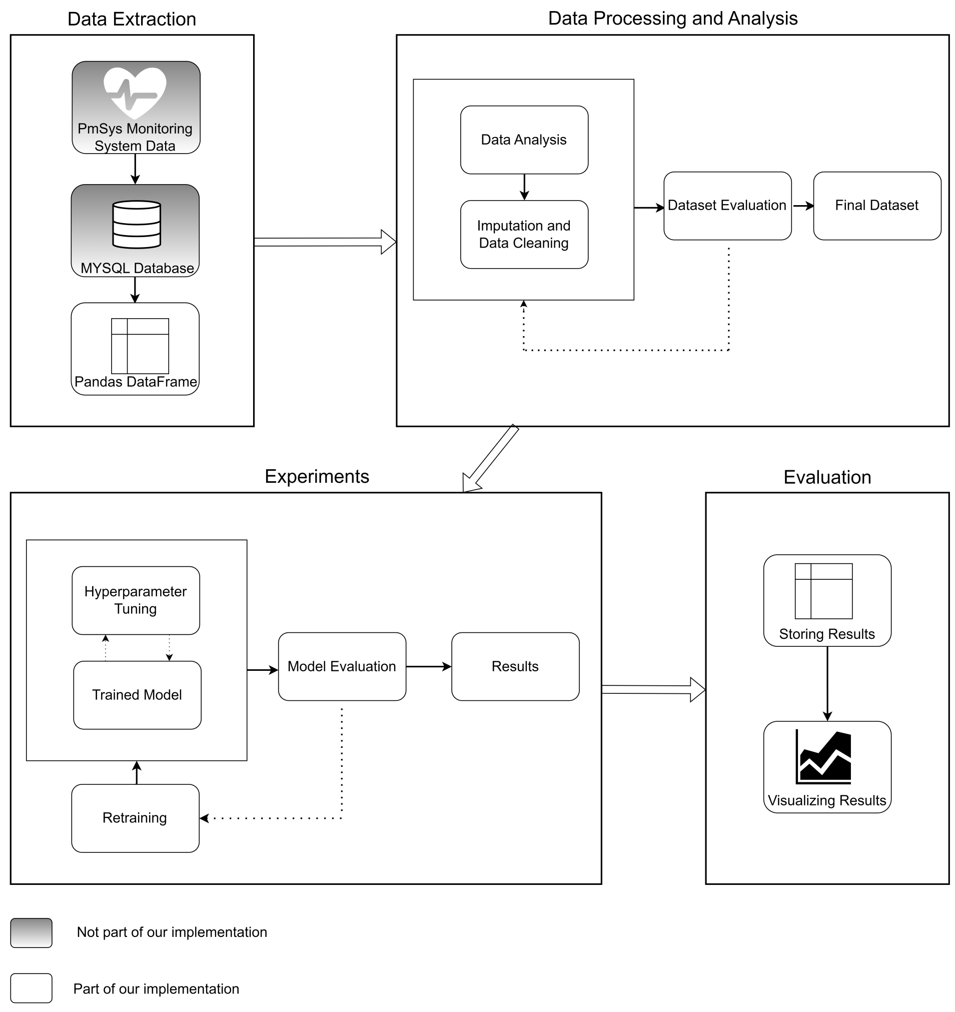


Figure 3.1: Illustration showing the workflow of each step in our proposed pipeline.

3.2.1 Data Importing

The process of importing data is the first step in our pipeline. We extract the data from an MySQL database. MySQL is a relational database management system that efficiently stores data in a table format [16], compatible with the time series data from the PmSys project [39]. We import specifically two tables containing subjective wellness and GPS-derived features. We have previously mentioned in chapter 2 section 2.6.3 regarding PmSys how each player submits their data through an app by answering short surveys. This data is subsequently stored in an MySQL database which we use. The GPS data is not gathered from the PmSys surveys but is generated from sensors worn by the players and then uploaded to the MySQL database.

3.2.2 Data Analysis and Processing

When the data is acquired, it is immediately processed to a format fit to be sent through our models. More specifically, the data is processed by several methods, such as imputation and handling of outliers. We use padding to fill all missing time steps with empty row values before removing all time steps before they started actively answering the survey and everything after they stopped answering the survey. This ensures that the days we are left with include only the period they actively answered the survey. However, this leaves us with missing data within the active periods, which is then imputed. Also, time steps containing wrongly formatted values or outliers are deleted or changed. Each player's active period is unique, resulting in time series data with variable lengths. We solved this issue by adding each player's data sequentially into our dataframe while incorporating time-specific features into the dataset. Also, our dataframe knows what dates are a result of imputation, so we can always choose between imputed and non-imputed data.

Be it holidays, rest days, injury, illness, or simply that they forgot to fill out the survey, all players, to some extent, have missing data. To combat this issue, we decided to impute these time stamps rather than accept them as missing. Our choice of imputation: IterativeImputer [50] is a multivariate imputer that works by estimating each missing feature value from all other features. An example of what that looks like can be found in Figure 3.2. The issue of imputation is that we stray farther from the actual data. As explained in Section 3.2.2, We have tried to minimize the number of data samples needed to be imputed by only including data from the period the players were actively answering the survey. However, many players still have many missing time steps within this period. The lack of actual data causes our final dataset to have 59% of its data directly resulting from imputation. In chapter 4, we will present the implications of performing this imputation on the dataset.

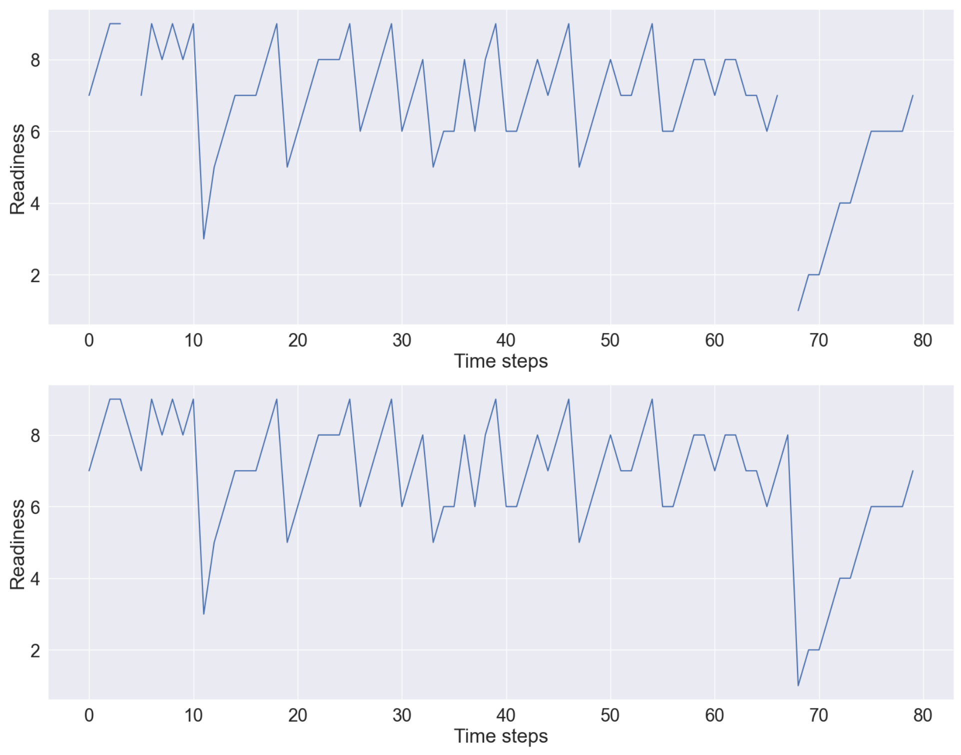


Figure 3.2: Images showing before and after using IterativeImputer on the data of one of the athletes in our dataset.

An important part of this thesis is understanding the data and its capabilities for time series forecasting. We, therefore, include a chapter dedicated to analyzing the statistical properties of the data, such as feature correlation, feature distribution, feature importance, stationarity, and how imputed data affects these statistical properties. As we will later see in Chapters 5, 6, and 7, understanding these statistical properties is instrumental in properly evaluating the performances of our experiments.

3.2.3 Experiments Overview

We use the processed data for all our different configurations of experiments where we produce different types of statistics. Some experiments create models for each model type for all players, while others only create models for each model type for one player. This is because we want a specific model for each player to maximize accuracy in a real-world scenario. Also, depending on the experiment, we want to see how the population of all players on a given team performs. At other times we only want to consider the performance of one individual player. Therefore, we produce results describing both the team and individual players.

We split our experiments into regression and classification sections for several reasons. Firstly, these two approaches are different, with distinct use-cases and limitations. Secondly, we want to clarify the different use-cases and how they differ from each other. Thirdly, these two approaches have a slightly different selection of models with different model configurations. Also, we are comparing how regression and classification contribute to the overarching research question. Therefore, separating these two approaches is sensible for both readability and clarity. Next, we will explain the reason for our model training schemes before looking at the different experiments and why we have included them.

It is important to understand how predictions in a time series domain work. We use the previous N days to predict the next M unknown days. The previous day(s) is the input window, while the next unknown day(s) is the output window. Figure 3.3 illustrates the process of a moving window procedure using the previous five days to predict the next three days.

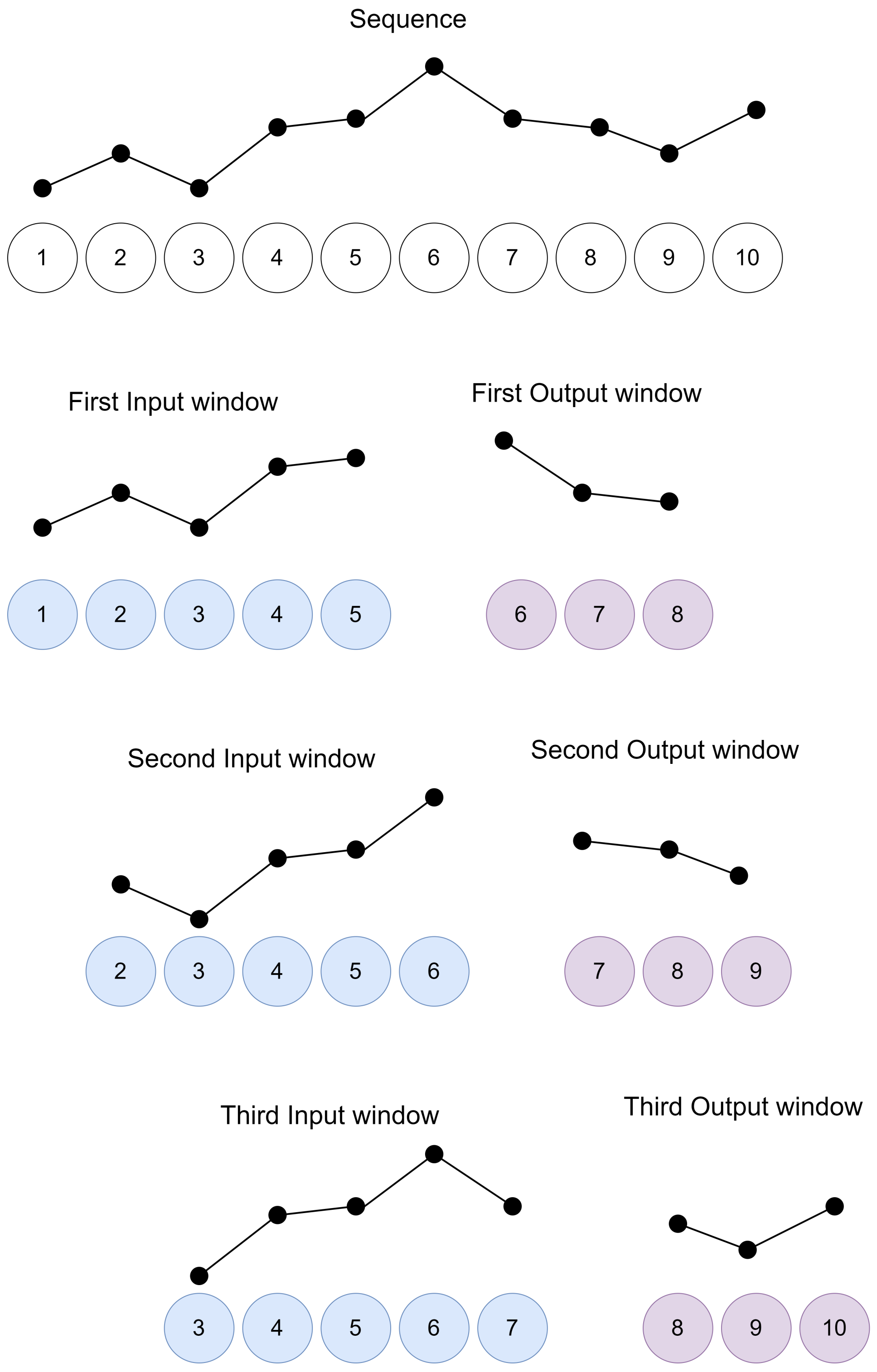


Figure 3.3: Illustration Showing The Process of Performing Sliding Window on Data.

Training Scheme

In chapter 2 in related work, we described how Wiik et al. [60] achieved the best results by training on the whole team's data rather than only using one player's data for both training and predicting. This is due to the lack of data for a new player joining the team. In the imputed dataset a player on average has 396 data points. For our pipeline, we intend to train using the whole team and predict for only one player. This method of training is more beneficial to the more complex deep learning models we are using since it allows us to use thousands of data points rather than only a few hundred if we were only to use data from one player at a time. Moreover, since the team dynamic also greatly affects each player's performance [60], training on the whole team might lead to discoveries of important trends. The instances where we look at all players in a given team we perform leave one out cross validation where each player is a sample. We will mainly look at the average value derived from the leave-one-out cross-validation. For all experiments, we will use team A except for the instances where we compare team A to team B. Reason for this is that team A has more data and less missing data in between the period the players actively answered the survey.

Hyperparameter Configuration

We are tuning hyperparameters for all relevant models to adequately represent each model's performance. The process of tuning the network parameters is done in several ways, but all methods are verified so they are not overfitting. For the TFT and LSTM models, we do a combination of manual tuning in combination with author and documentation recommendations. For the XGBoost model, we use a library called hyperopt [5] that automatically finds the best hyperparameters by leveraging an algorithm called Tree-based Parzen Estimators (TPE). The TPE approach outperforms Bayesian Optimization and Random Search [15]. The main perk of TPE is how it handles complex interactions among hyperparameters using a tree-based structure.

Regression Experiments

In our regression approach, we intend to answer questions regarding the optimal input window size, how large the forecasting horizon can be, what models among our selection are best suited to forecast readiness, the optimal forecasting method, what the most relevant features are, and how team A and team B compares in performance. These experiments will give important insight into the practicality of forecasting readiness and help answer our overarching research question.

- **Effect of Input and Output window sizes on RMSE-score:** In order to understand how the data parameters input and output window sizes affect the error rate, we decided to create plots showing the movement of the RMSE score when increasing the input window with several different output window sizes. This allows us to choose the optimal input window size. However, it is highly computationally costly to generate such plots.

- **Model Comparison Using All Features:** For each algorithm and player, we create unique models that all output an RMSE score. We use these values to create boxplots, one for each model, where each player represents one data point in the boxplot. This way, we can see the average error and spread of errors between all players on a team. We do this by using all features in our dataset
- **Model Comparison Using Only Readiness:** We perform the same experiment as the previous one mentioned, but only using the feature readiness instead of all features. The previous and current experiments let us compare if a multivariate approach provides less error than a univariate approach.
- **Incremental increase of output window using optimal input window size:** To see if it is possible to forecast for longer time horizons, we use the optimal input window size found in one of the previous experiments and try to forecast while increasing the output window with one up to 14. The motivation is to observe whether the error increases significantly from forecasting one day to N days.
- **One-Step Ahead versus Direct Forecasts:** By comparing different approaches to forecasting we can determine which method provides lowest model error.
- **Team A Versus Team B:** Comparing results forecasting on different teams might give insight into what data Characteristics are important on a population level when forecasting readiness to train.
- **Data Transferability:** By training on both teams we observe how transferable the data of one team is to another by evaluating the model predictions.

Classification Experiments

The purpose of conducting experiments through classification problems is to explore different ways of producing actionable statistics for coaches and players. Therefore, we have settled on three unique approaches to modeling readiness to provide actionable data.

- **Readiness- Forecasting 10 Classes:** As observed in table 4.1, we have that readiness is between 1 and 10. Therefore in a classification problem, we will have ten classes. The first classification approach will attempt to classify readiness to train, consisting of its ten original classes.
- **Readiness- Forecasting Peaks:** In section 4.4, we discuss how forecasting exact readiness values might not be important and that

forecasting peaks are a better approach. By this, we mean to forecast three classes: The first is all readiness values four and below, the second is five to seven, and the third is eight and above.

- **Readiness- Forecasting Positive, Negative Or Neutral Change:** Readiness is a subjective metric where the raw value of six might for one player represent a certain performance, while for another, it might be better or worse performance. Therefore, the actual usefulness of readiness might not be its numeric value; rather, the change in the readiness value might be more informative. Knowing if a player performs better or worse in the next game might be a more sensible interpretation of readiness than an arbitrary number.

3.2.4 Evaluation

The raw results produced from the experiments are in the format of a pandas dataframe, allowing the transformation of the data into any statistics. For this thesis, we produce a wide range of different plots to interpret the data easily.

3.3 System Specifications

All data analysis, model implementations, experiments, and evaluations are done within the Python programming language. Python has long been the go-to tool for data science tasks and offers extensive libraries for statistical analysis, as well as ML and deep learning models [13].

The main libraries used for our analysis and experiments is Pandas [38, 56], Scikit-learn [43], Darts [26], XGBoost [11], and pytorch [42]. All notable libraries used can be seen in table 3.1.

Name	Version	Description
Pandas	1.5.2	Dataset and analysis
Pytorch	1.12.1	Deep learning tool
Scikit-learn	1.2.1	Data processing and machine learning tool
CUDA	11.3.1	Required for running on GPU
mysql-connector-python	8.0.18	Used to extract data from mysql dataset
Statsmodels	0.13.5	Used to perform various statistical analysis
Darts	0.23.1	Deep learning library for time series forecasting
XGBoost	1.6.2	A boosted tree-based machine learning model
Sktime	0.13.2	Time series machine learning library

Table 3.1: Description of notable python libraries used.

Mysql-Connector-Python

The data used in this thesis is extracted from a MySQL database with the use of the Python library mysql-conncetor-python. The correct table is extracted and further converted into a pandas dataframe for preprocessing.

Pandas

Pandas [56] is an open-source library for python built on top of NumPY, that offers tools for data modeling, exploration, and analysis. It is highly used in data science for it's versatility in storing and manipulating data.

Scikit-Learn

Scikit-learn is a python library used for machine learning tasks. It offers algorithms and general purpose tools for machine learning.

Pytorch

Pytorch is an open source library that offers a framework used to create deep learning models. It is built on top of torch and is highly used in deep learning research.

Darts

Darts is an open source library for machine learning in the time series domain. The library offers a variety of models from RandomForest to deep learning networks such as the TFT.

3.4 Evaluation Metrics

This section introduces all evaluation metrics that are used to analyze the results generated in this thesis.

Mean Square Error (MSE)

Mean Square Error (MSE) is used in regression to measure error. It finds the average squared distance between the predicted and actual value. An MSE of zero means that the model has no error while MSE values above zero indicates model error with higher values indicating worse accuracy. One characteristic of MSE is that by squaring the values, it penalizes larger error more than smaller ones

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.1}$$

Root Mean Square Error (RMSE)

The metric Root Mean Square Error (RMSE) is similar to MSE because it is used to measure the average distance between target and actual value for regression problems. The main difference is that RMSE is measured in the same units as the response variable, while MSE is measured in squared units.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.2}$$

Confusion Matrix

A confusion matrix is used in classification problems to observe how well a model classify data to their respective classes. A confusion matrix with two classes, negative and positive, will look like that of Figure 3.4. Increasing number of classes will also increase the dimension size of the confusion matrix.

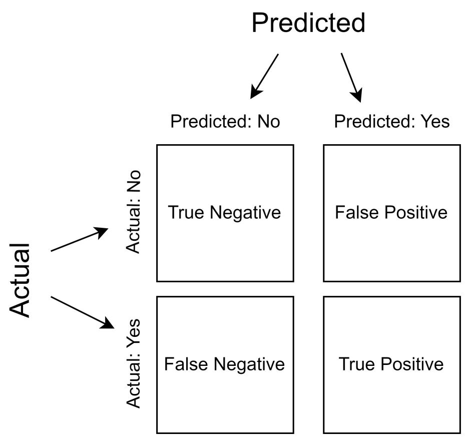


Figure 3.4: Example of a 2D Confusion matrix.

Accuracy

Accuracy is calculated by adding True Positive (TP) and True Negative (TN) and dividing by the number of total entries classified.

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

The accuracy score gives an exact measure of how many data points were correctly classified.

Precision

Precision gives the false positive rate, meaning that a model that produces no false positives, has a precision of 1. This is an important metric in the case were avoiding false positives is important and is calculated as follows.

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

Recall

Recall gives the true positive rate by offering a probability that a positive sample will be classified as positive. Recall is calculated as shown below.

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

F1-Score

The F1-score combines the two competing metrics, recall and and precision, to account for both FP and FN. The motivation is to have a single metric that optimizes recall and precision, which in itself is not possible since they work against each other. Therefore, in the case where both FP and FN are equally important to reduce, we choose F1-score.

$$\frac{2TP}{2TP + FP + FN} \tag{3.6}$$

3.5 Chapter Summary

This chapter presented our proposed pipeline to explore different use-cases and data configurations for forecasting readiness. The pipeline consists of a data extraction step, a pre-possessing and analyzing step, an experiment step, and an evaluation step. We described our system specifications providing the parts necessary to replicate our pipeline. Lastly, we introduced our evaluation metrics. The pipeline is relevant for both iterations regarding experiments seen in Chapter 5 and 6. The reason for the two iterations is to base the second iteration on what we learned from implementing the first set of experiments. Further, we intend to compare these iterations with each other. The motivation behind the two iterations is to provide important insight into our research question by exploring different ways of modeling time series data to forecast readiness and iteratively making improvements to our pipeline.

Chapter 4

Dataset Analysis and Preprocessing

The previous chapter described the overall methodology and how we intend to implement our pipeline. This included each step in our pipeline consisting of data importing, data processing, experiments, and presentation of results. We also explained why the chosen experiments are important to our research question. We provided the system specifications for our pipeline. and reviewed the evaluation metrics we will use for Chapters 5 and 6 regarding experiments.

In this chapter, we will discuss the properties of our dataset and why they are important to our forecasts, and how we process our dataset. The dataset comprises subjective wellness and GPS data with a target variable readiness to train and time-dependent variables such as the day of the week and month of the year. This chapter centers around understanding the data, where we analyze qualities such as distributions, correlations, and other important underlying factors when working with time series data.

4.1 Overview and Composition

The dataset contains features describing the overall wellness and training load of 45 players from two Norwegian professional female soccer teams. For simplicity, we will refer to these as teams A and B. Each time step represents one day, with the data collected from the start of 2020 until the end of 2021. However, given that some players did not join their respective teams until mid-season in either year, each player's range of data points varies. Therefore, only the data from which a player started using the PmSys monitoring tool and until the end of 2021 or they stopped using the PmSys monitoring tool is used. In other words, we only use the period when they actively recorded data. We chose to do this to ensure that the data, to a high degree, represents their actual wellness and performance, as it would make little sense to impute months of data for an arbitrary player from before they joined the team or after they stopped answering the survey. However, in between the active periods, missing data still exists, which is addressed later in this chapter.

We combine content from Tables 2.1 and 2.2 in addition to data from soccermon and GPS-derived[1] and time-specific features to create Table 4.1 representing our complete dataset. In Table 4.1, we see all the different variables with a description, data type, and value range. Compared to related work [34, 60], we have several more features available to thoroughly investigate the viability of using multivariate data for forecasting. Most notable is the inclusion of GPS-derived features. Further, we will determine the nature of our time series data and see whether it is stationary.

[1]GPS features have been derived from the raw SoccerMon data in collaboration with Lars Hoel (Master thesis: *Using Soccer Athlete GPS Monitoring Data to Visualize and Predict Features*, University of Oslo, 2023)

Metric	**Description**	**Data Type**	**Range**
Fatigue	The current fatigue level of the player	Numeric	1-5
Mood	The current mood of the player	Numeric	1-5
Readiness to train	The athlete's readiness for a training session or a game	Numeric	1-10
Sleep duration	The duration of the sleep	Numeric	0-12
Sleep quality	The quality of the sleep	Numeric	1-5
Soreness	The level of soreness	Numeric	1-5
Stress	The current stress level of the player	Numeric	1-5
Daily Load	Daily subjective load of a player	Numeric	0-
injury_ts	binary indication of injury with 1 suggesting an injury	Numeric	0-1
Total distance	The total distance run in one session (KM)	Numeric	0-
Average running speed	The average running speed for one session (m/s)	Numeric	0-
Top speed	Highest achieved speed in one session (m/s)	Numeric	0-
NumberOfHir	Number of high intensity runs	Numeric	0-
player name	Anonymous ID representing a player	string	None
Team name	Letter describing which team a given player belongs to	string	A-B
Month	The month of the year	Numeric	1-12
Day	The day of the week	Numeric	1-7
Date	The date specifying day, month, and year	Pandas datetime	01.01.20-31.12.21

Table 4.1: Description of features used in the final dataset for the first iteration.

4.2 Stationarity Test

An important trait to regard when dealing with time series is the concept of stationary data. For data to be stationary, the statistical properties of the time series can not change with time. This means that each data point in a time series is considered independent from each other. Further, no seasonal trend exists, meaning that the mean and variance in the entire time series are constant over time. The benefit of stationary data when doing time series analysis is that it is easier for the model to learn periodicity within the data. The issue with non-stationary data is that future data points might be several means higher than previous ones, which are difficult trends to learn.

An Augmented Dickey-Fuller test (ADF) [14] is used to investigate stationarity. The null hypothesis tests for a unit root in a time series, while the alternative test suggests that no unit root is present and that the data is stationary. The idea is that if we keep the null hypothesis, then the lagged value of the series (y_{t-1}) provides no useful observations in predicting (y_t) other than the lagged changes $(\triangle y_{t-k})$.

We implement the test using the library Statsmodels [54] with a max lag of 12. We perform the test for all individual player's data and reject the null hypothesis for all but a few of the players' data. There are abrupt shifts from the current trend for instances where we reject the null hypothesis. Because players comprise very few data points, a sudden shift in trend can cause the dickey-fuller test to treat some individuals as non-stationary. However, when treating all the players on the team as a complete time series, we can reject the null hypothesis and conclude that the data is stationary. Next, we examine feature correlations to determine their usefulness and possible trends.

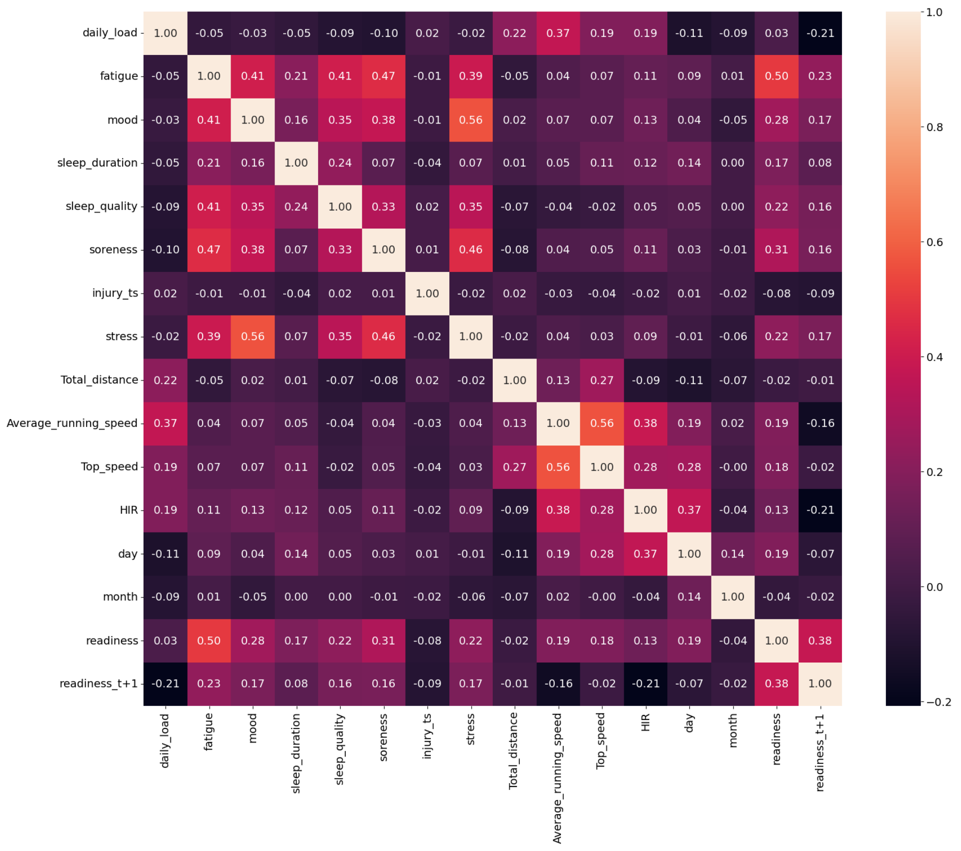
4.3 Feature Correlations

When dealing with ML, it is crucial to understand the data and the domain that data inhabits. Understanding the relationships between features makes it possible to understand how and why an increase or decrease in certain feature values affects other variables. In this initial analysis, we extract the most notable information from our correlation matrices as illustrated in Figures 4.1 and 4.2 where we show correlations for the unique teams' A and B.

We create the correlation matrices from the unprocessed dataset comprising 25 and 20 players for teams A and B, respectively. We chose to include the unprocessed dataset so that we can later compare it to the processed one to see if the data still has the same pattern and behavior. The correlation scores are the averages of all unique players for each team. Using the average among all players provides a more robust representation than generating the correlation matrix based on only one player. This is because individual players have few data points and have very different distributions.

Most notably, when observing the feature readiness in Figures 4.1 and 4.2, we see that wellness features like mood, soreness, and fatigue have much higher correlations to readiness than time or positional-features. However, it is reasonable to reason that readiness, a metric describing how ready an athlete feels for physical exertion, correlates more to an athlete's general wellness. It is also worth mentioning that all wellness metrics are answered in the same survey, which would explain some innate bias as to why they are more correlated to readiness than the objective features derived from GPS data.

In the correlation matrices, we also included a readiness feature one time-step in the future, denoted as *readiness_t+1*. The readiness variable itself is not our target variable. Rather, the target variable is readiness for one or more time steps in the future since we are forecasting time series data. For the correlation matrices of both teams, we see that all or some GPS features, especially HIR and Average running speed, have a high negative correlation to readiness_t+1. By comparing the correlation between GPS features and readiness_t+1, we observe that team B has a higher negative correlation than team A. Further, excluding the readiness and fatigue features, we see that HIR, Top_speed and Average running speed are the features most correlated to readiness_t+1 for team B. When looking at team A, we observe similarities only that Top speed has a very low correlation and that daily load has a high negative correlation. These results show that objective data, such as GPS-derived features, can have a greater impact on determining athletic performance, even perceived ones, than other wellness variables. Further, we will analyze the different distributions of our data, as well as how other features are affected by peak readiness values.



	daily_load	fatigue	mood	sleep_duration	sleep_quality	soreness	injury_ts	stress	Total_distance	Average_running_speed	Top_speed	HIR	day	month	readiness	readiness_t+1
daily_load	1.00	-0.05	-0.03	-0.05	-0.09	-0.10	0.02	-0.02	0.22	0.37	0.19	0.19	-0.11	-0.09	0.03	-0.21
fatigue	-0.05	1.00	0.41	0.21	0.41	0.47	-0.01	0.39	-0.05	0.04	0.07	0.11	0.09	0.01	0.50	0.23
mood	-0.03	0.41	1.00	0.16	0.35	0.38	-0.01	0.56	0.02	0.07	0.07	0.13	0.04	-0.05	0.28	0.17
sleep_duration	-0.05	0.21	0.16	1.00	0.24	0.07	-0.04	0.07	0.01	0.05	0.11	0.12	0.14	0.00	0.17	0.08
sleep_quality	-0.09	0.41	0.35	0.24	1.00	0.33	0.02	0.35	-0.07	-0.04	-0.02	0.05	0.05	0.00	0.22	0.16
soreness	-0.10	0.47	0.38	0.07	0.33	1.00	0.01	0.46	-0.08	0.04	0.05	0.11	0.03	-0.01	0.31	0.16
injury_ts	0.02	-0.01	-0.01	-0.04	0.02	0.01	1.00	-0.02	0.02	-0.03	-0.04	-0.02	0.01	-0.02	-0.08	-0.09
stress	-0.02	0.39	0.56	0.07	0.35	0.46	-0.02	1.00	-0.02	0.04	0.03	0.09	-0.01	-0.06	0.22	0.17
Total_distance	0.22	-0.05	0.02	0.01	-0.07	-0.08	0.02	-0.02	1.00	0.13	0.27	-0.09	-0.11	-0.07	-0.02	-0.01
Average_running_speed	0.37	0.04	0.07	0.05	-0.04	0.04	-0.03	0.04	0.13	1.00	0.56	0.38	0.19	0.02	0.19	-0.16
Top_speed	0.19	0.07	0.07	0.11	-0.02	0.05	-0.04	0.03	0.27	0.56	1.00	0.28	0.28	-0.00	0.18	-0.02
HIR	0.19	0.11	0.13	0.12	0.05	0.11	-0.02	0.09	-0.09	0.38	0.28	1.00	0.37	-0.04	0.13	-0.21
day	-0.11	0.09	0.04	0.14	0.05	0.03	0.01	-0.01	-0.11	0.19	0.28	0.37	1.00	0.14	0.19	-0.07
month	-0.09	0.01	-0.05	0.00	0.00	-0.01	-0.02	-0.06	-0.07	0.02	-0.00	-0.04	0.14	1.00	-0.04	-0.02
readiness	0.03	0.50	0.28	0.17	0.22	0.31	-0.08	0.22	-0.02	0.19	0.18	0.13	0.19	-0.04	1.00	0.38
readiness_t+1	-0.21	0.23	0.17	0.08	0.16	0.16	-0.09	0.17	-0.01	-0.16	-0.02	-0.21	-0.07	-0.02	0.38	1.00

Figure 4.1: Pairwise correlation matrix of our data before imputation for Team A.

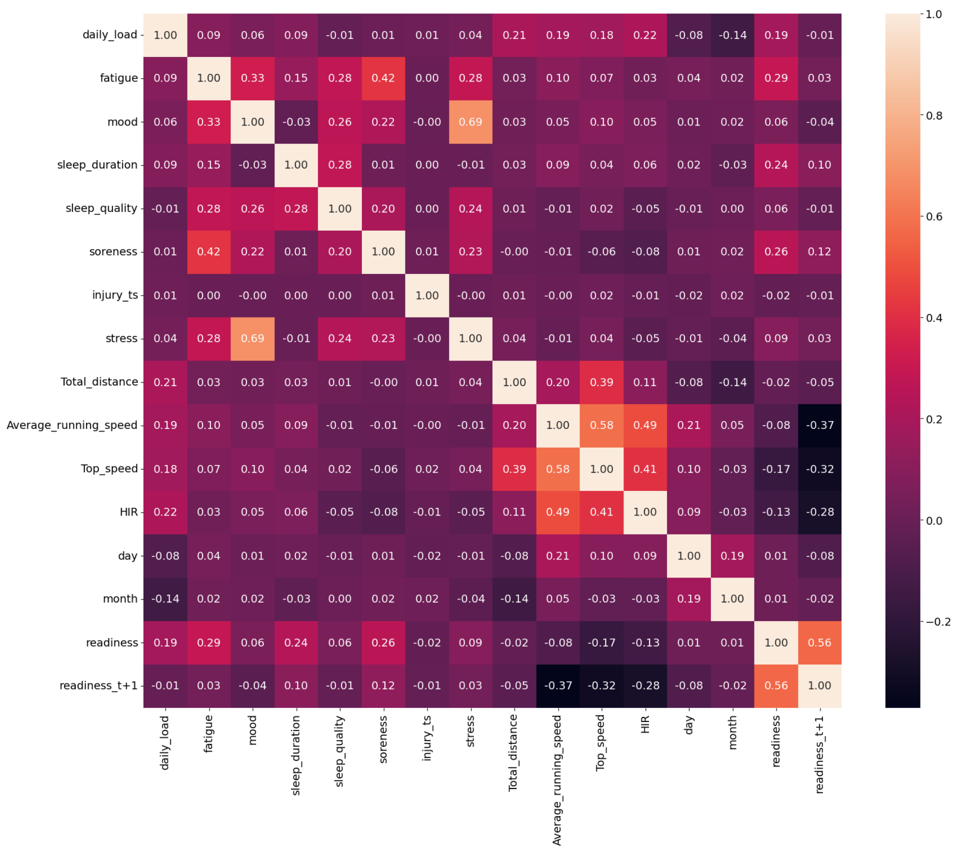


Figure 4.2: Pairwise correlation matrix of our data before imputation for Team B.

4.4 Data Distributions

For our experiments, we are doing both regression and classification. Therefore, understanding the distributions within the data is useful for both use-cases, although more important for classification problems. The issue lies with skewed distributions. If 80% of the data belongs to a single class, and the chosen model has an 82% accuracy, then the model would not really be better than always choosing the majority class. The purpose of this section is to describe the underlying distribution of the variable readiness and how the distribution of other variables is affected by readiness.

4.4.1 Overview

Readiness is the target value, so it is important to know its distribution to determine how well our models perform. Knowing how well the models perform relative to each other is not enough. This is why we will use dummy models for model comparisons in Chapters 5 and 6. From Figure 4.3, we can see clear differences between the different teams, with team A consisting of more extreme values and generally having data points more evenly spread. One important thing to note is that most data points center around values six, seven, and eight. We expect that athletes mostly find themselves at an adequate level of readiness, as low values of readiness would suggest high levels of stress, fatigue, soreness, or an injury, which should be a rare occurrence. On the other hand, a high readiness score would suggest peak performance relative to their current average physical state, which should be rare. However, the values suggesting fatigue or peak performance are most useful to map since they impact an athlete's athletic state the most.

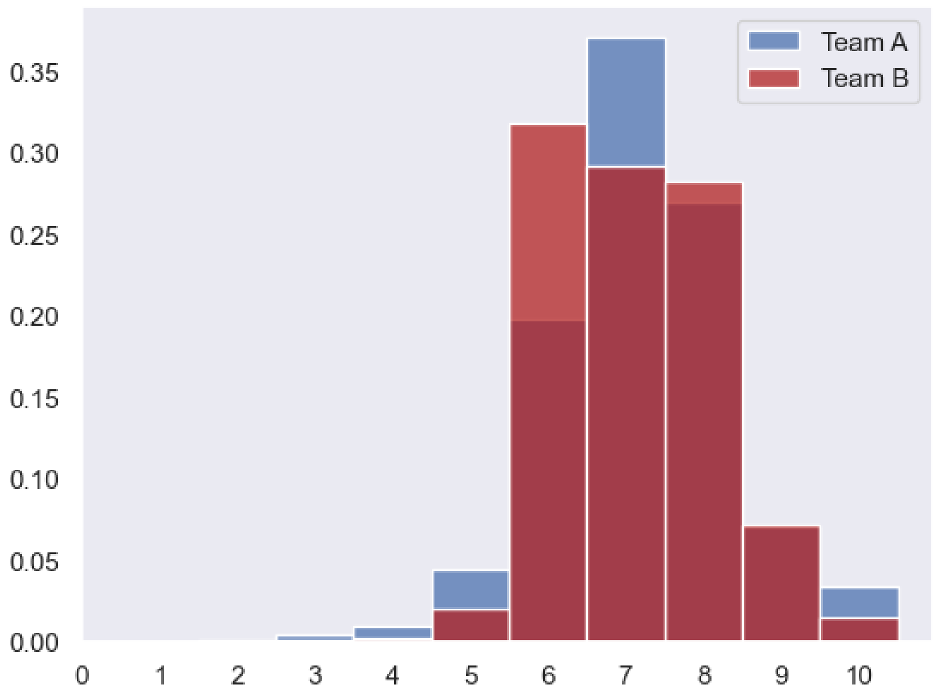


Figure 4.3: Histogram showing distribution of readiness for both Team A and Team B. X-axis denoting the readiness score, while y-axis denotes the percentage between zero and one.

The practical application of forecasting degrees of readiness is limited since most distribution lies between good and very good readiness values. Predicting data within this range is not necessarily useful information for clubs, as what will have the most impact is forecasting the edge cases when readiness is very low, signaling fatigue or possible injury, and very high, suggesting peak performance. Everything in between suggests an average perceived athletic state. Predicting small trend changes might prove useful when athletes are subject to changes in how they train. In that case, observing how the readiness of players or teams moves after incorporating changes might lead to greater insight into athletic development.

	Team A	Team B
Mean	7.18	7.1
Median	7	7
Standard Deviation	1.17	1.06
Readiness <5	1.4%	0.29%
Readiness Between 5 And 7	61.25%	62.97%
Readiness >7	37.34%	36.74%

Table 4.2: Table showing numeric values describing the distribution of readiness for Team A and Team B before imputation.

In Table 4.2, we see numerical values describing both teams' statistical properties of readiness. We observe that both teams have an almost identical mean. Further, the overall grouped distribution of readiness seen in the last three rows in Table 4.2 are also very similar. Although, with a notable exception, team A has a considerably higher representation of readiness values that are four and below. Team A also has a higher standard deviation resulting in a distribution farther from the mean than Team B. This observation is important because team B will likely have less model error when performing regression tasks and higher accuracy when doing classification tasks purely based on the data distribution.

4.4.2 Individual Players

The general wellness among individual players can vary substantially and is apparent when viewing the histograms in Figure 4.4. Some players might have almost all occurrences in a single score, while others have readiness scores more evenly spread over multiple values. Skewed distribution among players is important to consider, as what can be considered good accuracy scores for individual players will vary depending on their distribution.

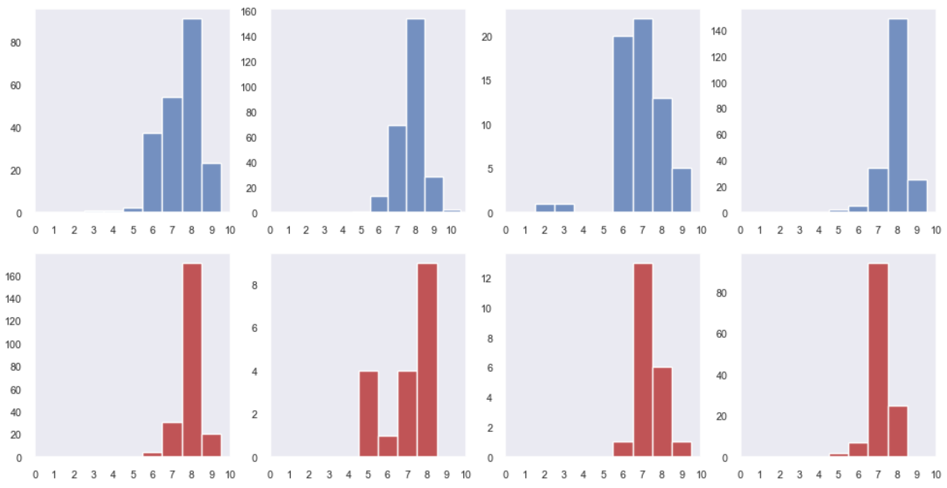


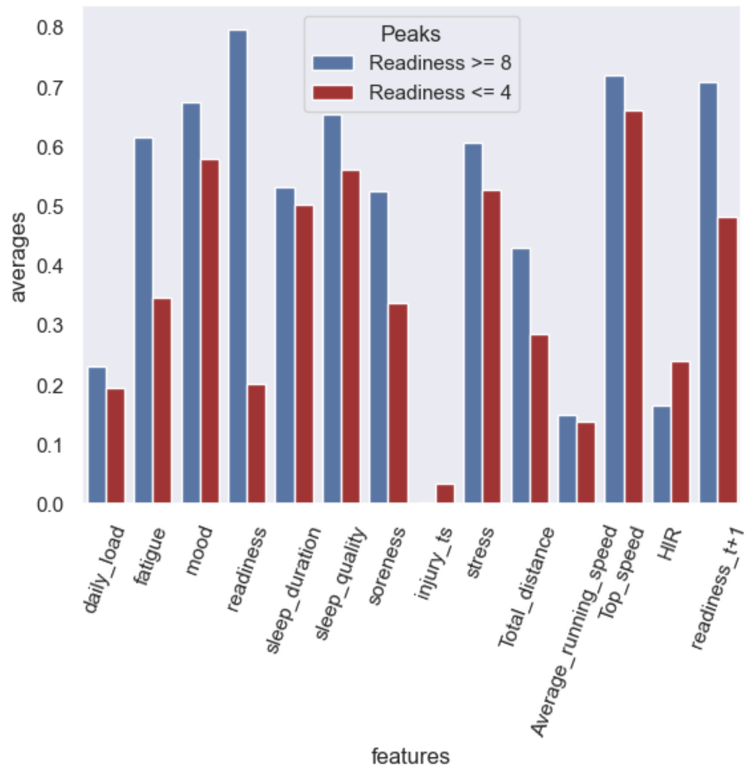
Figure 4.4: Histograms showing distribution of readiness for eight players, four from both Team A (blue plots) and Team B (red plots). X-axis denoting the readiness score, while y-axis denotes the number of occurrences.

4.4.3 Characteristics of Peak Readiness

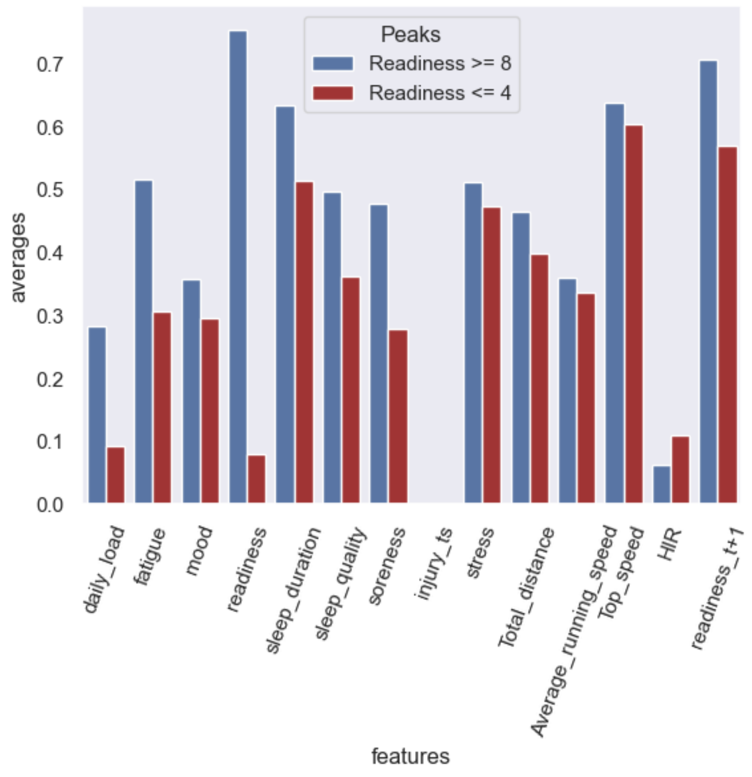
As discussed in Section 4.4, the extreme values are most important for clubs to identify and understand. Understanding how other metrics behave when readiness is either very high or very low could potentially indicate important trends and allow for a better understanding of what dictates perceived athletic performance.

In Figure 4.5, barplots show the distribution of features when readiness is four or lower and eight or higher to see if there are any major differences in feature distributions when readiness is very low or very high. Unsurprisingly, lower readiness values also result in lower wellness values, such as fatigue, soreness, and sleep quality. Other wellness values such as stress, mood, and sleep duration are also lower but only around 5-10%. Further, observing the GPS-derived features shows that HIR is consistently much higher when readiness is low. However, the total distance is much lower, with the average running and top speeds being the same. Generally, this suggests that HIR is a good indicator of readiness. We also included the readiness value for the next time-step to see the average recovery readiness value after experiencing low readiness. We see that for the next session, readiness, on average, goes quickly up to higher values of readiness. We also see that team B recovers faster than team A. Daily load is lower when readiness is four and below compared to eight and above. Team B especially sees much lower values for daily load than team A. Team B only has one case of injury rendering it useless. In contrast, team A has more than 20, enough to extract useful information. By looking at team A in Figure 4.5, we see that injuries are prevalent when readiness is low compared to high.

The results from Figure 4.5 are promising in regards to using multivariate data to forecast readiness since they show consistent differences in several feature values among low and high readiness values for both teams. Especially the wellness values fatigue, soreness, sleep quality, and GPS-features such as HIR and total distance.



(a) Team A



(b) Team B

Figure 4.5: Barplots showing the distributions of features when readiness is four or below and eight or higher.

4.5 Imputation

In Chapter 3 Section 3.2, we explained our imputation method. We chose to use IterativeImputer [50] because it supports imputation for both discrete and continuous variables and is intended for multivariate data [50]. The method is also far more complex than previous methods used in related work [34, 60]. However, imputation is an enormous field on its own [40], and other methods might prove more robust for our work. Still, our aim for imputation is to use an adequate method to demonstrate its potential for these kinds of data. Next, we will describe the missing data and compare the statistical properties of the imputed dataset with the dataset before imputation.

4.5.1 Statistics on Missing Data

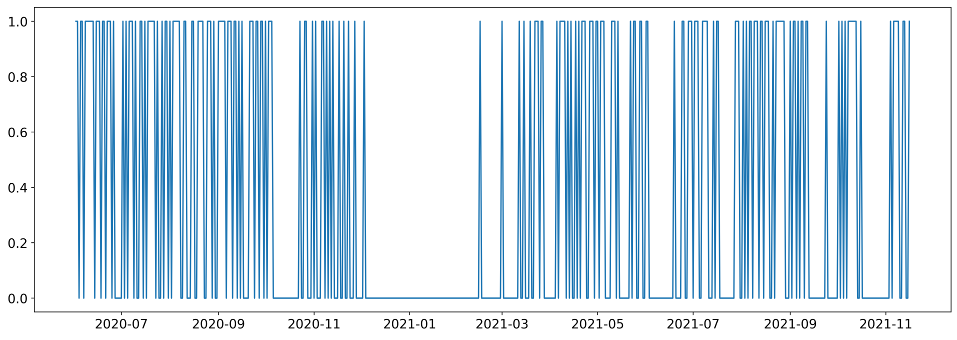


Figure 4.6: Binary plot showing missing dates for a random player. A value of zero indicates a missing value.

The plot in Figure 4.6 shows when missing data occurs for a random player. The most notable observation is how many dates containing missing data are clustered together. Especially the dates from late December 2020 to March 2021 are void of data. All players in our dataset have this two to three-month absence of data around that time. Several of the players also share the same smaller clusters of missing data. These findings suggest that much of the missing data among players is, in fact, not entirely random. However, the possibility that professional soccer players have a period where they do not train for three months is highly unrealistic. The probability that they did not use wearable monitoring tools for some reason in that period is more likely, especially since these periods are common among players. Further, most occurrences of missing dates only consist of one missing day. However, in Figure 4.6, larger clusters of missing data with a size of seven or greater account for 57% of all the missing values.

4.5.2 Correlation

For team B we observe in Figure 9.2 that the correlations between features and readiness after imputation are very similar to those observed in the dataset before imputation in Figure 4.2. The correlation between wellness parameters is around the same, with fatigue, sleep duration, and soreness having the highest correlation. As for GPS-derived features, we see that for team B, top speed, HIR, and average running speed still substantially correlate to readiness, as the data did before imputation. Overall, Team B is not greatly affected by performing imputation, with the previous most correlated features still having the highest correlation after imputation. Also, comparing readiness_t+1 we observe the same development. As for Team A, in Figure 9.1, we observe that the GPS features correlate less to readiness_t+1 than the dataset before imputation. However, comparing readiness before and after imputation for team A, we generally see the same distribution of correlation among all features.

It is promising that the overall relationship between features is intact, albeit with feature correlation deviating by a few percentages. However, some deviation is expected, especially since the imputation process replaced up to 59% of a given player's data.

4.5.3 Distribution

By observing the histogram in Figure 4.7 we see that the distribution is slightly different from Figure 4.3 representing the data before imputation. The main differences in the imputed dataset are the increased number of very low values for both teams and a reduction in very high values. This difference is greater in team A. Since we have fewer very high values, more are between five and seven. However, the standard deviation of both teams has increased, which we observe from Table 4.3. Increased numbers of very low readiness values and a higher standard deviation indicate that the ML models will experience more error using the imputed dataset.

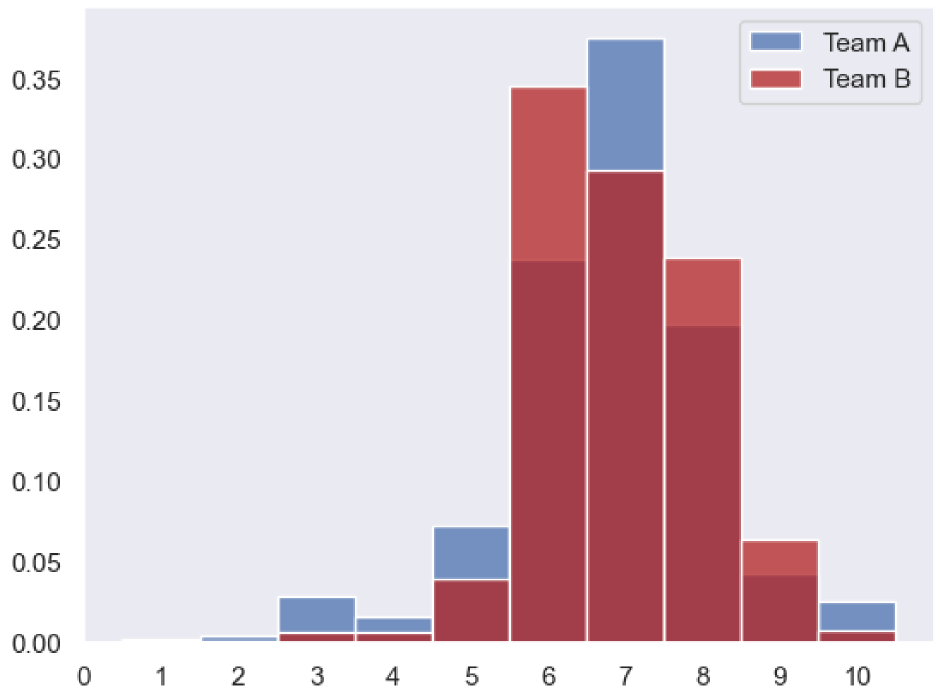


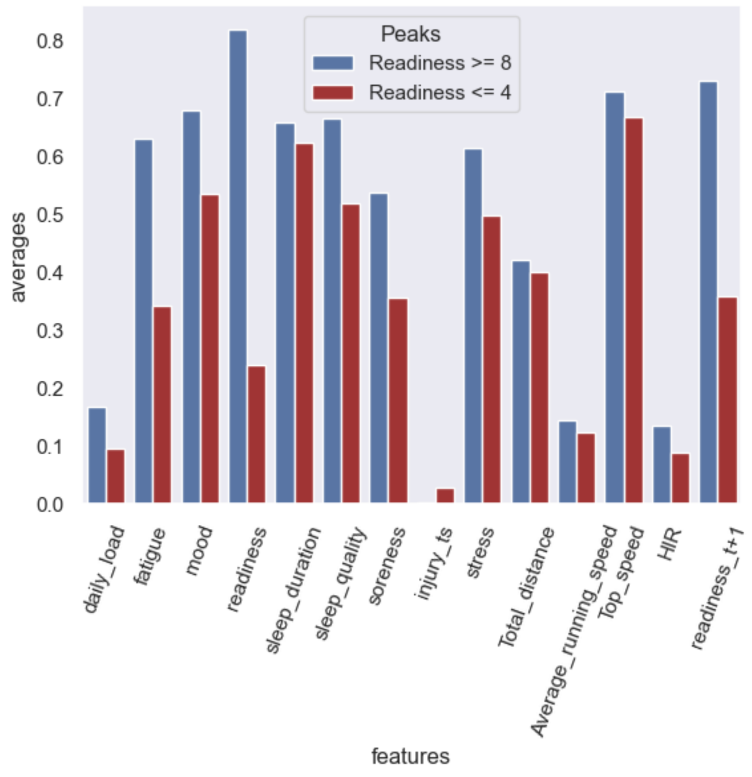
Figure 4.7: Histogram showing distribution of readiness for both Team A and Team B after imputation. X-axis denoting the readiness score, while y-axis denotes the percentage between zero and one.

Table 4.3 tells us that the average readiness value in the imputed dataset is slightly decreased by a few percentages. Generally, team B is less affected by the imputation and retains more statistical properties from before using imputation compared to team A.

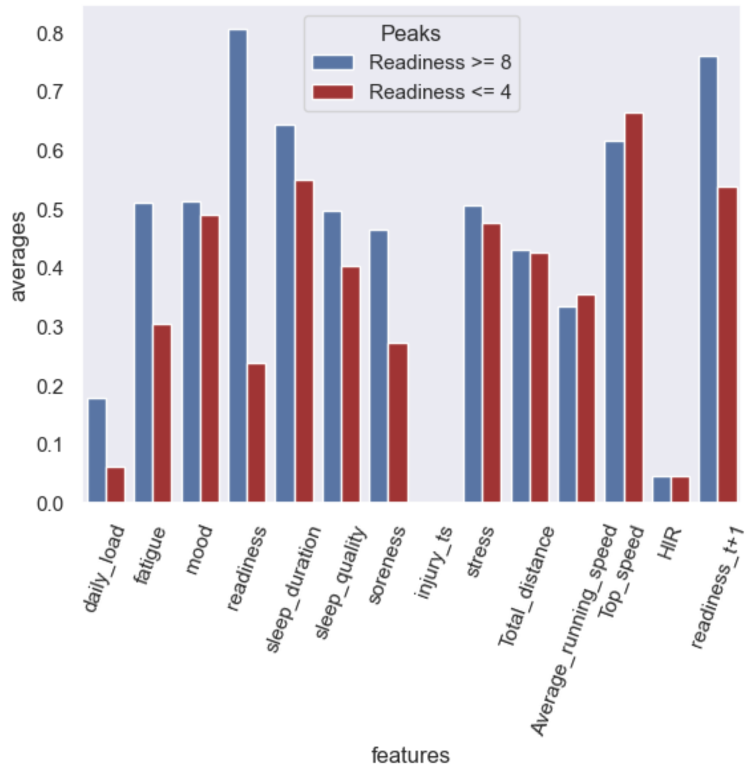
	Team A	Team B
Mean	6.78	6.91
Median	7	7
Standard Deviation	1.36	1.13
Readiness <5	5.09%	1.4%
Readiness Between 5 And 7	68.41%	67.63%
Readiness >7	26.5%	30.97%

Table 4.3: Table showing numeric values describing the distribution of readiness for Team A and Team B after imputation.

We observe Figure 4.8. The wellness parameters fatigue, soreness, and sleep quality have the same distribution pattern. However, The GPS data, especially HIR and total distance, are now either equal for both cases of very low and high readiness or have been skewed. This is possible to observe for team A regarding HIR, where very high readiness on average has a higher HIR value. Before imputation, we observed the opposite.



(a) Team A



(b) Team B

Figure 4.8: Barplots showing the distributions of features when readiness is four or below and eight or higher after imputation.

4.5.4 Notable Changes to the Dataset

Generally, we observe that imputation did not change the statistical properties of the dataset to such a degree that it no longer represents the players. However, it is still a less true representation of the players nonetheless. A higher standard deviation and an increase in very low readiness values will likely increase model error. Although, more examples of very low readiness values might help the model better learn such trends. Overall, the correlation between features and the distribution of readiness stayed roughly the same.

4.6 Feature Importance

Our correlation matrices already indicate what features are important in predicting readiness. Nonetheless, to derive more certain conclusions, we need further evidence. Therefore, this section will generate permutation feature importance scores and SHAP values to determine how features impact our predictions.

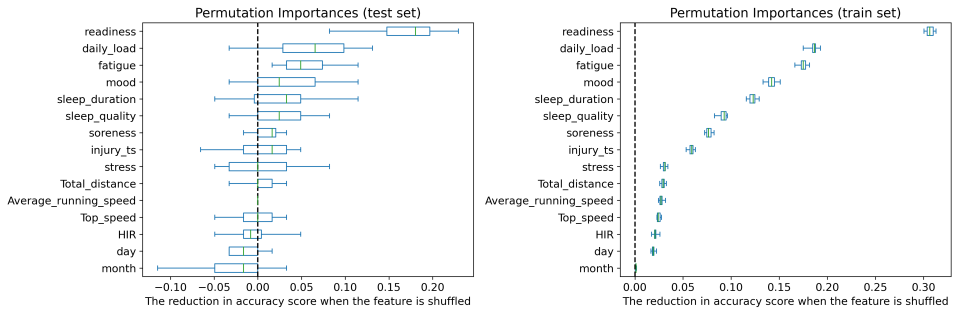


Figure 4.9: Permutation feature importance plots.

Permutation feature importance works by answering the question: What effect on accuracy occurs when the data of a single feature is randomly shuffled in the validation set while all other data remains the same? A prediction using the non-shuffled data is first obtained and then compared to the prediction using shuffled data. The permutation feature importance score is the difference between these two predictions. This process is repeated to create scores for all features. In Figure 4.9, we have the permutation importance scores for all features for both the training and test set. We observe that all features in the training set are, to some extent, important. In the test set, we observe that all the GPS-derived and time-related features are inconsequential to the prediction. The negative scores indicate that the random shuffled values had more impact on predictions than the real values. Apart from the lagged readiness value, the daily load is the most important feature, followed by the remaining wellness features except for stress. The correlation matrices earlier in this chapter suggested that GPS-derived features had a good correlation to readiness. However, in this test, GPS-derived features do not impact model output.

A possible answer to the behavior of the permutation method is that the GPS-derived features are highly correlated, meaning that when one feature is shuffled, that feature is still available to the model through other GPS-derived features. This causes reduced feature importance scores for the correlated features.

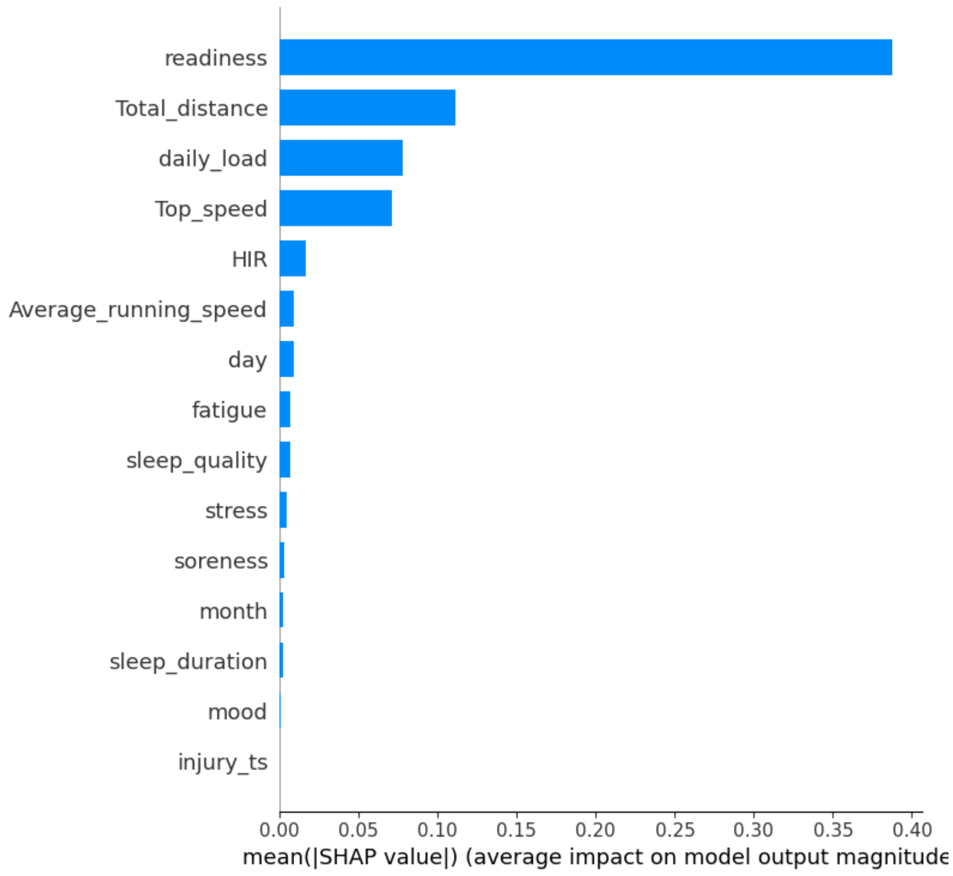


Figure 4.10: SHAP Plot.

SHAP values, an acronym from SHapley Additive exPlanations [37], introduces a method to understand the importance of each feature in a prediction. It is based on cooperative game theory, and the process can be thought of as a game with multiple players where each player's contribution (or feature) is given based on the model output. We visualize each feature's average importance on predictions in Figure 4.10. Interestingly, we use the same model and data configurations we used for the permutation feature importance plots. We observe the opposite results where wellness features have little to no impact on model predictions while GPS-derived features have a very high impact on model predictions.

4.7 Chapter Summary

In this chapter, we presented the underlying statistical properties of our dataset and discussed their significance regarding model forecasting. These properties include stationarity, feature correlations, readiness and other feature distributions, and feature importance. We also gave a comparison between the imputed data and the non-imputed data. We showed that much of the statistical properties remained the same despite the impact of imputing data.

Chapter 5

Experiments and Results - First Iteration

In the previous chapter, we analyzed our dataset and described its statistical properties. The purpose of this chapter is to present the results derived from our experiments for the first iteration. We split our results into two sections, one for regression and another for classification. For each section, we first provide the parameter configuration for all models. Further, we go through the results of each experiment and explain their importance to our research question. At the end of this chapter, we summarize our findings.

5.1 Regression Models

This section will present our findings, forecasting with five different regression models. The models used in this section are **LSTM**, **XGBoost**, **Linear Regression**, **Decision Tree**, and **TFT**. Moreover, this section will thoroughly describe the model configurations and discuss general thoughts about the models and their behavior during training. We then proceed sequentially to go through all experiments and present their results.

5.1.1 Model Parameters

To better understand the more subtle mechanisms of the models and make it possible to reproduce all experiments, it is necessary to discuss and provide the hyperparameters used for each model. All models use a random seed to maintain reproducibility, meaning the experiments are possible to replicate while producing the same result. Hyperparameter tuning is not the focus of this thesis but is done to represent each model and its performance adequately.

Linear Regression- Configuration

We are using Sklearn's ordinary least squares Linear Regression model. We included this basic model to observe how other more complex models perform against it. Since we are using several features and many with high correlations between each other, the linear regression model might be prone to overfitting. However, we tested the model with several train/val/test sets combinations and saw good fits for all data splits.

Decision Tree Regressor- Configuration

The Decision Tree regressor, along with the linear regression model, are fairly simple models. The idea is also to compare the decision tree model with the more complex models.

Long Short-Term Memory- Configuration

The parameter configuration of the LSTM model can be viewed in table 5.1. The parameters were manually tested and set, and the number for epoch and batch size is a good compromise between performance and computational speed. Further, early stopping with patience has been implemented. Early stopping is a method used to stop training when the validation loss no longer decreases. Patience is the number of epochs the model will run without measuring any improved validation loss. If early stopping with patience of two is applied, the model will stop training after observing two consecutive epochs with no improved validation loss. Applying early stopping is important since we want to reduce validation loss as much as possible, not training loss. By not including this method, we end up overfitting our model to the training data.

Type	Hyperparameter	Value
Data	Batch Size	16
	Epoch	12
	Input Sequence Length	7
	Output Sequence Length	1
Network	Learning Rate	0.001
	Number Of Hidden Layers	32
	Loss Function	RMSE
	Number of LSTM layers	2
	Optimizer	Adam
	Dropout	0
	Early stop	0.005

Table 5.1: LSTM- Hyperparameter configuration.

eXtreme Gradient Boosting- Configuration

The XGBoost model has the following parameters as shown in table 5.2. The model parameters were found using the open-source hyperopt library [5] as explained in 3.2.3 where we utilize the TPE algorithm. We also applied early stopping for this model.

Type	Hyperparameter	Value
Data	Input Sequence Length	7
	Output Sequence Length	1
Network	Learning Rate	0.01
	booster	'gbtree'
	n-estimator	150
	objective	'reg:squarederror'
	colsample_bytree	0.97
	max_depth	9
	min_child_weight	3.47
	reg_lambda	0.45
	seed	0

Table 5.2: XGBoost- Hyperparameter configuration.

There was little or no difference in using the default parameters compared to the tuned hyperparameters. Using different train/test/val splits, we observed only a few percentages reduced loss at most.

Temporal Fusion Transformer- Configuration

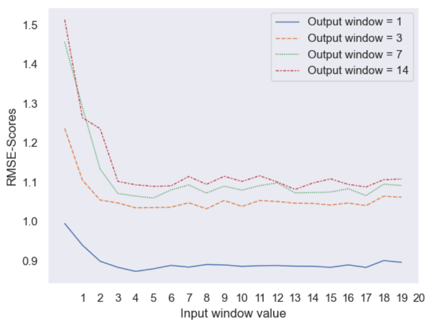
The hyperparameters of the TFT model were found by manually testing different configurations. An increase in batch size led to much faster computation and less accurate results, which meant more epochs were needed. Therefore, we struck a balance to optimize performance and results. Further, early stopping is applied to stop training when validation loss is no longer shrinking over a period of 5 epochs. We used a patience value of 5 since, during training; validation loss would often go up substantially before decreasing again. All other features were the default values recommended by the library darts [25].

Type	Hyperparameter	Value
Data	Batch Size	32
	Epoch	40
	Input Sequence Length	7
	Output Sequence Length	1
Network	Learning Rate	0.001
	Number Of Hidden Layers	64
	number of attention heads	4
	Number of LSTM layers	1
	likelihood	QuantileRegression
	Dropout	0.1
	Optimizer	Adam
	Early stop	0.005

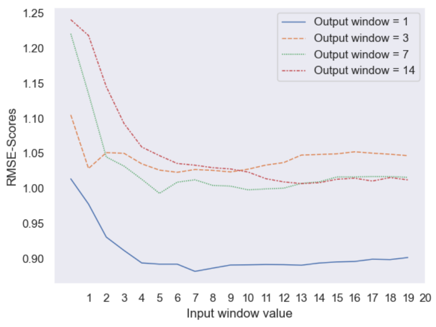
Table 5.3: TFT- Hyperparameter configuration.

5.1.2 Size of Input and Output Windows

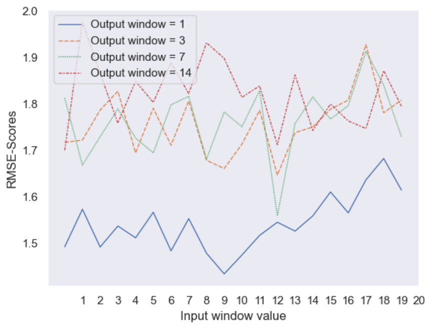
One of our sub-questions is investigating the optimal input window size, which means we need to understand the relationship between past and future time series. In order to properly answer this question, it is necessary to see how our models perform given different input but also output window values. Figure 5.1 shows five plots, each a unique model, and illustrates how input window size affects RMSE scores. Each data point for each model is the RMSE value of training on team A and forecasting a specific player with a unique input and output window values configuration. Therefore, the plots show how input and output window values affect the forecasting error for a single player. For this experiment, we limited the predictions to only one player since creating these plots even for the non-deep ML algorithm XGBoost demanded a lot of computational resources. The particular player we forecast consists of 502 data points and has a baseline RMSE of 1.22 when using a model only predicting the mean. We chose this particular player because the player is among the ones with the least missing data. The player also has a high spread in readiness distribution with several very high and very low readiness values relative to the other players. The plots also have four lineplots with different output window values. This is to see if forecasting time steps further in the future changes the optimal number of input window values.



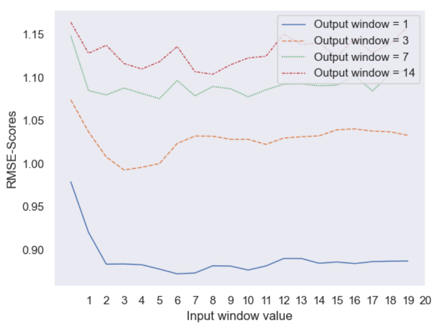
(a) XGBoost



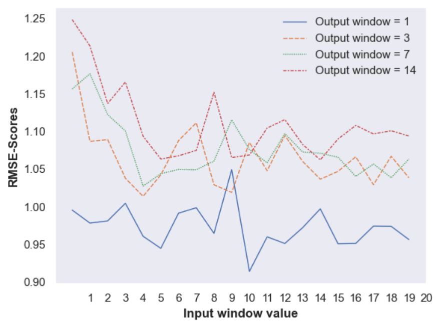
(b) Linear Regression



(c) Decision Tree



(d) LSTM



(e) TFT

Figure 5.1: Five plots showing five unique models on how different input and output window values affect the RMSE-score. The plot starts at x-axis = 0 which has an input window of one.

Figure 5.1 shows that input window size larger than one produces less error. The models XGBoost, Linear Regression, and LSTM have very similar behaviors, where each of the four lineplots in each sub-figure goes drastically down for the first few increases of input window size. From there, the lineplots find a minimum point, and the error increases again before a plateau is met. The most decrease in RMSE happens when the input window value is one and increases to three. The model TFT and the decision tree do not follow the same trend. For the decision tree, it does not seem like the input window size has any meaningful impact on error. For the TFT model, we see at least for larger output window sizes that the error consistently goes down with the first few increases in input window size. However, there is no obvious trend when the output window value is one. The minimum point is the optimal input window size, and for these three models, we observe that it occurs between values five and eight. We also observe for the three best models that the optimal input window size is around the same for all output window sizes. We see a clear trend for the three models achieving the lowest error: LSTM, XGBoost, and Linear Regression have an optimal input window value between five and eight days.

It is also important to point out that the plots in Figure 5.1, indicates that the RMSE scores are much higher for output window sizes above one compared to RMSE scores when the output window is one. Also, the difference in RMSE scores between configurations with higher output window values is either very low or non-existent. In other words, an increase in output window size after one increases RMSE, but an increase in output window size when the output window size is more than one does not increase RMSE by substantial levels. It might seem positive that RMSE does not increase when the output window size goes from three to seven or fourteen. However, the large increase of RMSE when increasing the output window from one suggests that forecasting multiple days into the future is difficult for the models to learn. Figure 5.1 shows a substantial increase in RMSE, only increasing the output window size from one to three. In some cases, configurations with a higher output window size might have a lower RMSE score than those with a smaller output window size. These findings support the notion that the forecasts using larger output windows are too uncertain to extract useful data.

The plots in Figure 5.1 offers a unique perspective on how input and output windows affect forecasting of readiness to train and indicate that the most recent days are by far the most important. By increasing the output window, we did not see any major trend difference for the input window. For the remainder of this chapter, we will use seven as the value for the input window since it, on average, produces the least error among the best-performing models. Further, we discuss the issue of forecasting longer time horizons in Section 5.1.3.

5.1.3 Forecasting Horizons

A part of our research question is to investigate how large the forecasting horizon can be before the model output becomes too unreliable. In Section 5.1.2, we briefly examined how the RMSE score went up substantially after only increasing the forecasting horizon from one to three. We also observed that the RMSE score did not change much, going from seven to fourteen, suggesting that the models beyond a certain value for the output window only forecast the mean rather than following a reasonable trend. Further in this section, we will show how much the error increases for all models when we increase the forecasting horizon to discover when the uncertainty in the model outputs becomes too high.

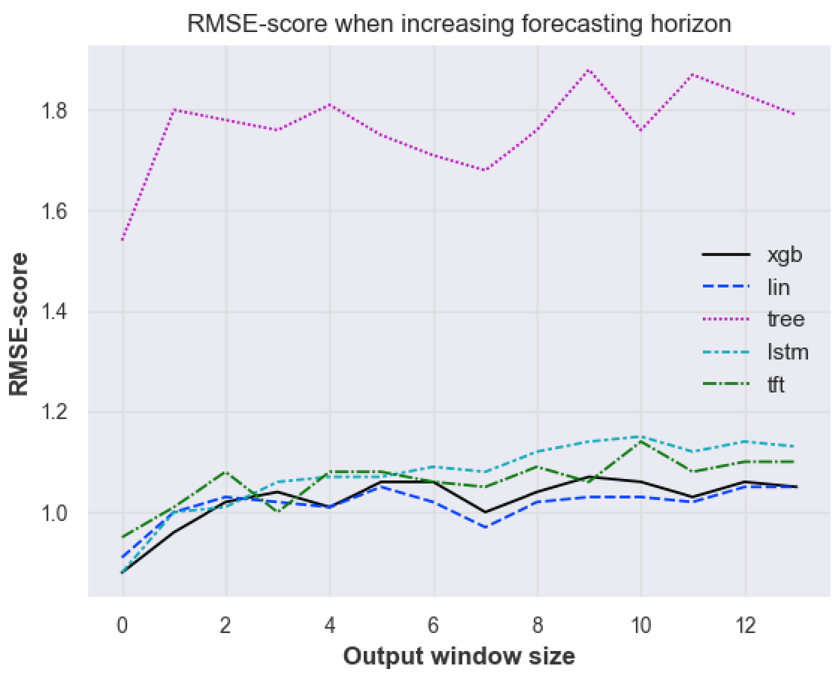


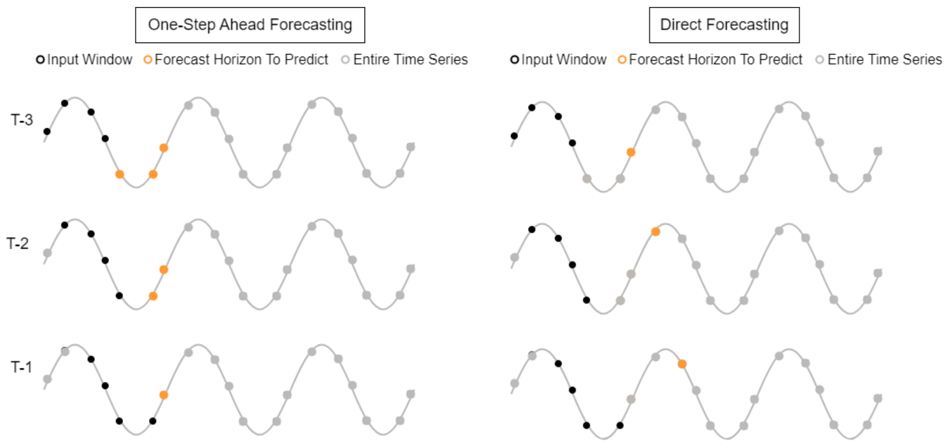
Figure 5.2: The Line-plot shows how the RMSE-score of all the models is affected by increasing the forecasting horizon. X-axis value equal to zero represents an output window of one

To see how an increase in forecasting horizon affects each model, we created the plots in Figure 5.2. The lineplots use the data from team A, and each output window size configuration forecasts the readiness of the same player for all models. A clear trend for all models is apparent, with RMSE scores rising for each increase in the forecasting horizon until the output window is three. The difference in RMSE scores is almost negligible for most models when going from three to fourteen. These findings suggest that forecasting readiness with this approach for longer time horizons is too difficult for the models to provide useful forecasts and that the models perform as well with a forecasting horizon of three as it does fourteen. Generally, we see that reliably forecasting several time steps into the future is not attainable with our approach.

By observing the performance of each model from Figure 5.2, we see that XGBoost and linear regression has the least error throughout. At the same time, the LSTM starts with a lower RMSE score than the TFT, but the LSTM ends up with a larger error after the output window equals six. The tree regressor performs considerably worse than all the other models. It offers no useful knowledge other than reinforcing the concept that increasing the output window from one increases the error. None of the models deviated in terms of the trend where an output window larger than one drastically increased the model error. These findings show that no specific model among our selection can forecast longer than one time step without drastically increasing the RMSE. Therefore, except where mentioned, we will use an output window of one.

5.1.4 Recursive Multi-step Versus Direct Forecasts

Another point of interest is to investigate whether the technique of recursive multi-step forecasts is better than direct forecasts. By direct forecasts, we mean a model only trained to forecast a specific time-step interval that can be any N number of time-steps. On the other hand, one-step ahead forecasts will reuse the predicted value(s) as part of the input to make the prediction for the next time step and recursively repeat this process for each unknown time step. An illustration visualizing both methods is available in Figure 5.3.



(a) Direct Forecasting Method

Figure 5.3: Image to the right visualizes how the direct forecasting method works by training the model to (in this case) forecast values three time-steps beyond the input data. Image to the left shows the one-step ahead forecasting method that retains the same input window but for each next time-step will reuse it's prediction in it's input window to generate the next prediction.

To observe potential differences in error between one-step ahead and direct forecasts, we created the barplots shown in Figure 5.4. The figure shows four sets of barplots, each with a specific configuration of forecasting horizon and forecasting method. The forecasts use data from team A and forecasts for a single player. We make comparisons between the barplots with the same output window sizes. Left to right, we observe that the

RMSE score between XGBoost and Linear Regression for the first two sets of barplots with an output window equal to three are the same. We observe the same occurrence for the next pair of barplots with an output window of seven. Further, the decision tree has a higher error when using the direct method with an output window of three. However, the opposite happens when the output window is seven, and we observe a lower error using the direct method. As previously stated, the simple decision tree is seemingly not able to understand the data reliably. These plots show that we can not justify any difference between these two forecasting methods.

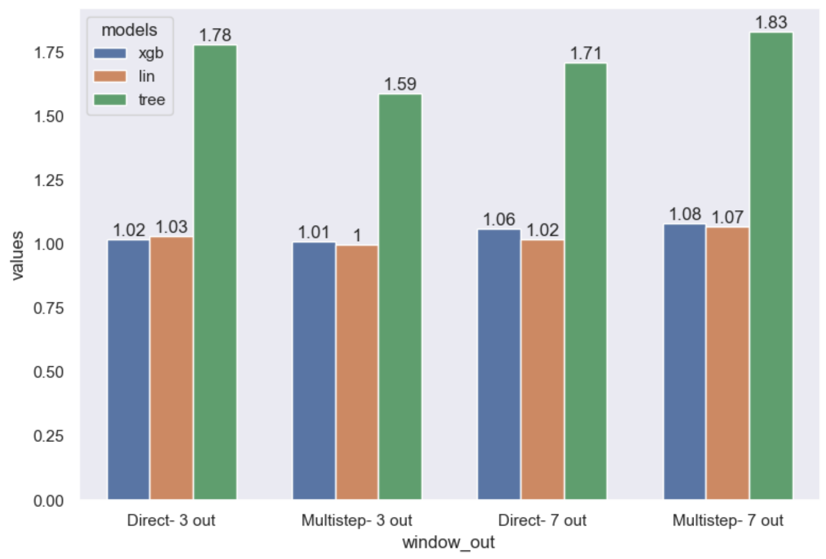


Figure 5.4: Barplots showing the difference in RMSE-scores using either direct or one-step ahead forecasts. This example uses the XGBoost, linear regression, and tree regressor model.

5.1.5 Multivariate Versus Univariate Data

One of our sub-questions, important in answering our main research question, is to investigate whether the forecasts become more reliable by adopting a multivariate approach to the data. Rather than only using the lagged value of readiness, we can incorporate several other variables that also correlate to the target variable. The addition of new variables might make the forecasts more accurate. Therefore, this experiment aims to provide a comparison of how the models perform using both multivariate and univariate data.
To understand how the models perform across all players in a team, we created the boxplots in Figures 5.5 and 5.6. The idea is to see how each model performs when training on the team and forecasting readiness for a specific player. Previously mentioned literature found that this method yielded the most accurate results [60]. The player used to forecast is not

in the training set, as that would be a blatant example of memory leakage. Rather, in our case, we are using the data from team A, which consists of 25 players, meaning we are training 24 players and forecasting one. We do this for all players on team A, meaning we are performing leave-one-out cross-validation. The motivation behind this is to see the spread in forecasts of each model when used in a real-world scenario where we want to forecast the performance of all players.

5.1.6 Multivariate Data

The boxplots in Figure 5.5 and the table in Figure 5.4 describe how the models perform forecasting each unique player in the population team A using multivariate data. The XGBoost, Linear Regression, LSTM, and TFT model yields the most accurate results, with the TFT having a slightly worse performance. Comparatively, the tree regressor performs considerably worse. The best four models also have about the same distribution of outliers. The tree model is the worst performer and seems to have difficulties learning the different trends in the data.

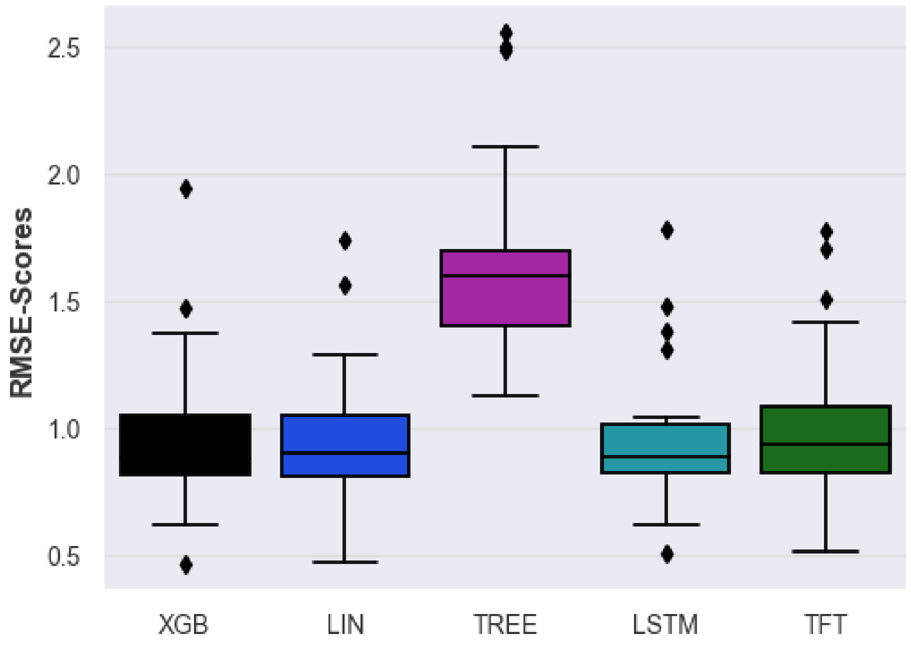


Figure 5.5: The illustration shows RMSE-Scores expressed through boxplots. Each data point in the boxplot is the RMSE-Score of predicting all readiness values for a single player, with all other players being used for training. In this case we have used team A and predicted RMSE-Scores for all players, meaning 25 data points for each boxplot.

Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	0.97	0.95	1.67	0.95	1	1.32
Median	0.88	0.9	1.6	0.89	0.94	1.13
SD	0.32	0.28	0.39	0.29	0.32	0.52
Min	0.47	0.47	1.13	0.51	0.52	0.45
Max	1.94	1.74	2.56	1.78	1.77	2.86

Table 5.4: The table describes the boxplots in Figure 5.5 with numerical values. We also included the values from a dummy model only predicting the mean.

Looking further at Table 5.4 and the column describing the results from the dummy model, we can compare the ML models and the dummy model. The dummy model only predicts the mean, which means that if the other models do worse or not significantly better, they will not be better than just predicting the average value of readiness. We see that the tree model performs far worse than the dummy model. Whereas the other models, on average, have around 40% less error than the dummy model. However, if we look at the median values, we see that the four best-performing models only do about 20% better than the dummy model. A high discrepancy between the mean and the median values suggests that the dummy models have several forecasts of players with very high RMSE scores. These are data with high variance in which forecasting the mean yields a very high error.

Further observing min, max, and standard deviation values in Tables 5.4 and 5.5, shows that the ML models have a much lower spread than the dummy model. This is attributed to the ML models attempting to learn patterns while the dummy model only predicts a straight line. Therefore, the dummy model performs worse when the data has a high variance. The best performing ML models produces increasingly better results than the dummy model when the variance in the data increases. We tested the effect variance has on the dummy model in Figure 9.8, and it shows a significant relationship where an increase in variance also increases model error. We also perform t-tests on the best-performing models and the dummy model with a p-value threshold of 0.05 and observe that the best-performing models have significantly less error. Moreover, we previously mentioned in section 4.4 that the data distribution is mostly centered around the same values, meaning that for some players' data, a good strategy is to predict the mean.

5.1.7 Univariate Data

The same type of boxplots and table describing the overall error over different train/test splits explained in section 5.1.6 was also created using only the readiness variable and is shown in Figure 5.6 and Table 5.5. We see the same model behavior observed in Section 5.1.6 with XGBoost, Linear

Regression, and LSTM achieving the best performance. TFT performs slightly worse, and the decision tree performs considerably worse.

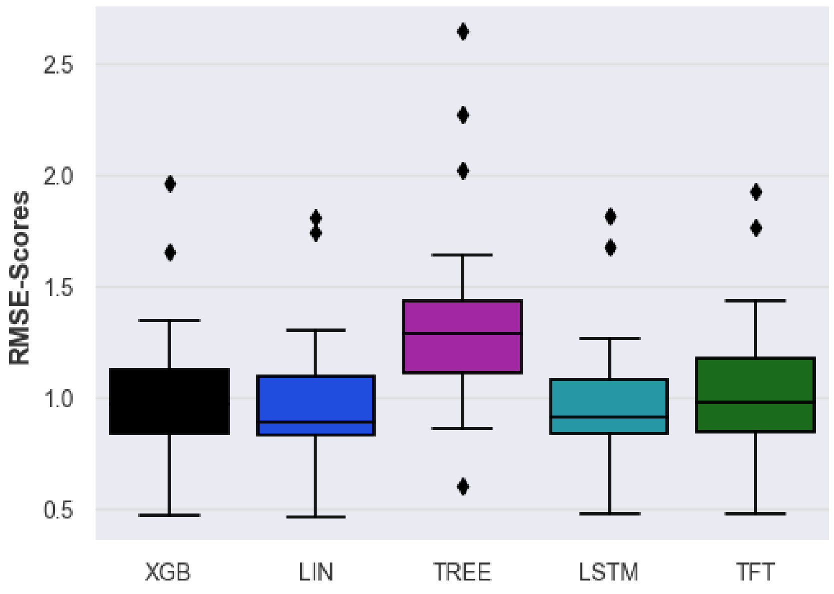


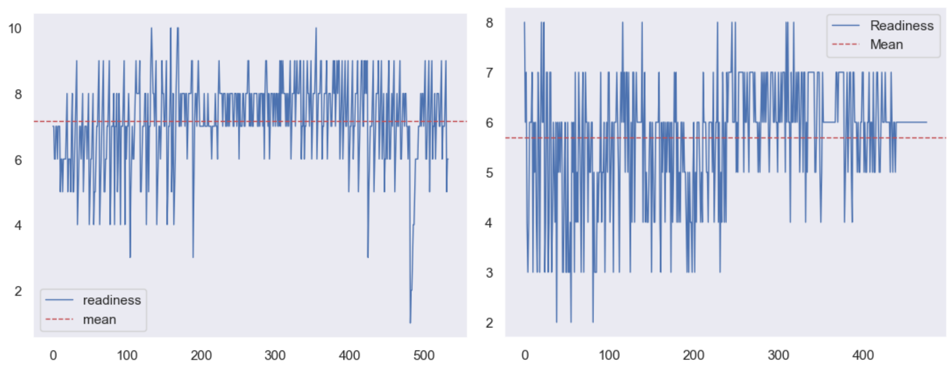
Figure 5.6: The illustration shows RMSE-Scores expressed through boxplots as explained in Figure 5.5 but only using univaraite data.

Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	1	0.99	1.35	0.99	1.04	1.32
Median	0.97	0.89	1.28	0.91	0.98	1.13
SD	0.32	0.31	0.44	0.3	0.33	0.52
Min	0.47	0.47	0.6	0.48	0.48	0.45
Max	1.96	1.81	2.65	1.82	1.93	2.86

Table 5.5: The table describes the boxplots in Figure 5.6 with numerical values. We also included the values from a dummy model only predicting the mean.

5.1.8 Comparison of Multivariate and Univariate Data

By observing the numeric values such as mean and median in Tables 5.5 and 5.4, it is apparent that a multivariate data approach in most cases has a positive effect on model output. Multivariate data performs better on average for all models except for the tree regressor. The variance is around the same, if not lower, for multivariate data models. Generally, these preliminary results suggest that adding more relevant variables gives the models more context on to base their forecasts and reduces error. However, the difference is small, with XGBoost having the largest improvement with 9% less median error using a multivariate approach. For the Linear Regression, LSTM, and TFT, we see an improvement of -1.7%, 2.3%, and 4% less median error using a multivariate approach, respectively. Still, we see a much higher discrepancy between using multivariate and univariate data when looking at individual players rather than a team's average.



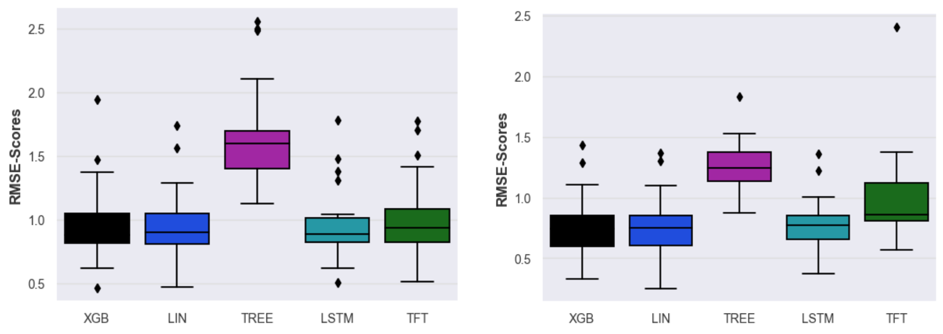
(a) Player 1 (b) Player 2

Figure 5.7: Two plots where Player 1 represents the player having the best performance using multivariate data and Player 2 having the worst performance on multivariate data compared to using univariate data.

Figure 5.7 displays readiness values from two players. Players 1 and 2 are the examples with the greatest gap in error between using multivariate and univariate data. Player 1, when forecasting with a multivariate approach, performs 11% better than using only univariate data, and Player 2 performs 9% better using a univariate approach over a multivariate one. The difference in variance is not high, with Player 1 having a variance of 1.75 while Player 2 has a variance of 1.46. In this case, we see no obvious reason as to why one player performs much better with multivariate data while another performs much better with univariate data. We performed a linear regression analysis with difference in RMSE value between using multivariate and univariate data as the target variable and variance as the explainable variable. The analysis is available in Figure 9.7. We can reject the null hypothesis by observing the T-test with a p-value threshold of 0.05. The analysis suggests no significant relationship, meaning variance does not determine when multivariate data positively or negatively affect predictions.

5.1.9 Team A versus Team B

To observe possible discrepancies between team A and team B, we conduct the same experiments done in Section 5.1.5 using team B's data. This lets us see how our models perform on both teams.



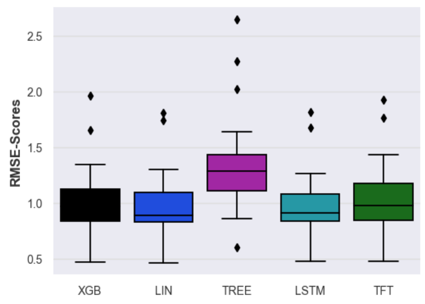
(a) Team A (b) Team B

Figure 5.8: The figure shows two images, each using the data from either team A or team B using all features.

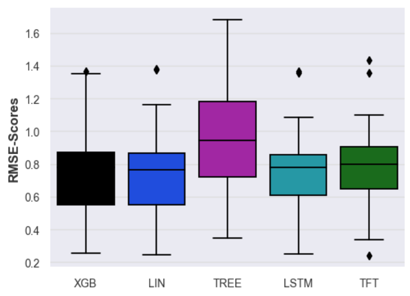
Team A- Multivariate						
Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	0.97	0.95	1.67	0.95	1	1.32
Median	0.88	0.9	1.6	0.89	0.94	1.13
SD	0.32	0.28	0.39	0.29	0.32	0.52
Min	0.47	0.47	1.13	0.51	0.52	0.45
Max	1.94	1.74	2.56	1.78	1.77	2.86
Team B- Multivariate						
Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	0.76	0.75	1.25	0.77	0.99	1.05
Median	0.74	0.75	1.24	0.77	0.86	1.05
SD	0.28	0.27	0.23	0.24	0.4	0.28
Min	0.33	0.25	0.87	0.38	0.57	0.6
Max	1.43	1.367	1.83	1.36	2.41	1.84

Table 5.6: The table describes the boxplots in Figure 5.8 with numerical values. We also included the values from a dummy model only predicting the mean.

Looking at both the boxplots in Figure 5.8 and the statistics in Table 5.6, we observe that team B has a less overall error. However, the behavior of the models is approximately the same. The decision tree performs considerably worse than all other models, and the TFT is slightly worse than the remaining models. The LSTM, XGBoost, and Linear Regression models perform best and obtain roughly the same RMSE score. Further, we learned from Section 4.4 that team B has less variance and outliers, meaning forecasting values closer to the mean result in lower RMSE values than team A.



(a) Team A



(b) Team B

Figure 5.9: The figure shows two images, each using the data from either team A or team B using univaraite data.

Team A- Multivariate						
Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	1	0.99	1.35	0.99	1.04	1.32
Median	0.97	0.89	1.28	0.91	0.98	1.13
SD	0.32	0.31	0.44	0.3	0.33	0.52
Min	0.47	0.47	0.6	0.48	0.48	0.45
Max	1.96	1.81	2.65	1.82	1.93	2.86
Team B- Multivariate						
Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	0.76	0.75	1.25	0.77	0.99	1.05
Median	0.74	0.75	1.24	0.77	0.86	1.05
SD	0.28	0.27	0.23	0.24	0.4	0.28
Min	0.33	0.25	0.87	0.38	0.57	0.6
Max	1.43	1.367	1.83	1.36	2.41	1.84

Table 5.7: The table describes the boxplots in Figure 5.9 with numerical values. We also included the values from a dummy model only predicting the mean.

We compare A with B when fitting univariate models. Again, team B has a lower RMSE, and the model behavior is the same between the teams. The decision tree and the TFT perform much better using univariate data. The difference between using multivariate over univariate data is less in team B. Team A benefits more from using multivariate data. We attribute this to team B having less variance than team A, shown in Figure 4.2. Therefore, team B is less affected by wrongly forecasting peak values since fewer exist.

5.1.10 Data Transferability

We investigate whether the unique dynamics within a team are important when forecasting readiness for a player. When playing together, the idea is that all players on a given team will impact each other's performance.

Further, this means that the data among players on a team should represent the data from an arbitrary player from that team better than an arbitrary player from another team. However, it would be beneficial to clubs if it is possible to use data from one team to train models to predict player data from other teams since these data types are sparse and hard to obtain. Therefore, we test whether using the data from both teams for training positively affects the results.

We observe the Figures 9.3, 9.4, 9.5, and 9.6. For team A, we observed no improvement or worse accuracy depending on the model for both univariate and multivariate data. As for team B, we saw no improvement except when using the TFT, which managed to achieve 30% less error using univariate data training on both teams. However, this was the only example where training using both teams led to better results. Further, we attribute this to TFT being a complex model needing massive data to function properly [35].

5.2 Classification Models

An important aspect of our research is exploring different approaches to forecast readiness to train. As previously mentioned in Section 4.4, it might be more beneficial for athletes and coaches to use classified readiness values because discrete values are more intuitive to interpret than continuous ones. Also, with a classification approach, it is possible to make the problem easier for the models by reducing the number of classes. Our selection of models for classification is **Ridge Classifier**, **ROCKET**, **LSTM**, and **XGBoost**. We chose not to include a decision tree classifier based on the poor performance of the decision tree in Section 5.1.5. The TFT is also not included based on its poor performance. We chose to include ROCKET because of its state-of-the-art results on classification datasets [12]. We will also compare these models to a dummy model that only predicts the majority class. Next, we will describe three use-cases for classification using our dataset.

For our experiments, we have used three different approaches to classify readiness. The aim is to provide useful statistics for both players and clubs. The first approach is classifying the readiness variable using the original ten classes. The second approach is similar to the first but uses only three classes to forecast peaks. Lastly, the purpose of the third approach is to forecast the trend. Rather than interpreting an exact readiness value, the model indicates a positive, negative, or neutral change in the readiness score. The models need to output actionable data, so it is important to explore several different use-cases. Next, we will declare the model parameters.

5.2.1 Model Parameters

We declare the model parameters configured for each model used in our classification approach.

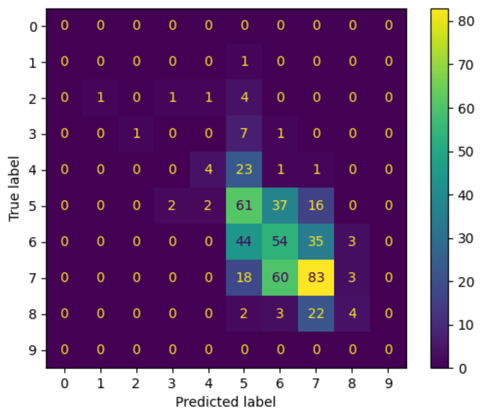
- **Linear Classification- Configuration:** For our Linear classifier model, we are using Sklearn's Ridge classifier model. We use the default parameters [43].
- **ROCKET- Configuration:** To implement the rocket model, we use the recommended model parameters described in the paper [12]. We set the kernel size to 10 000 and the alpha for the ridge classifier to logspace(-3, 3, 10).
- **XGBoost- Configuration:** We tested different hyperparameters with hyperopt but found that reusing the same parameters from Section 5.1 yielded the best results.
- **Long-Short-Term Memory- Configuration:** We tested different hyperparameters manually but found that reusing the same parameters from Section 5.1 yielded the best results.

5.2.2 Classification Experiment Results

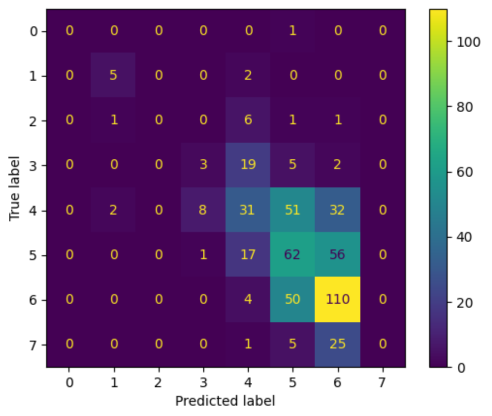
Next, we will go through each of the three use-cases. We use accuracy, F1-scores, and confusion matrices to evaluate the results. Particularly, F1-scores are used since the data in our classification experiments has a high class imbalance. The input window is set to seven with a forecasting horizon of one. Because each player has widely different data distributions, using a player with readiness values distributed across all classes as the test set is reasonable. We will use the player from Section 5.1. In addition to having fewer missing values than most other players, this player also has several very low and very high readiness values.

Forecasting The Original 10 Classes

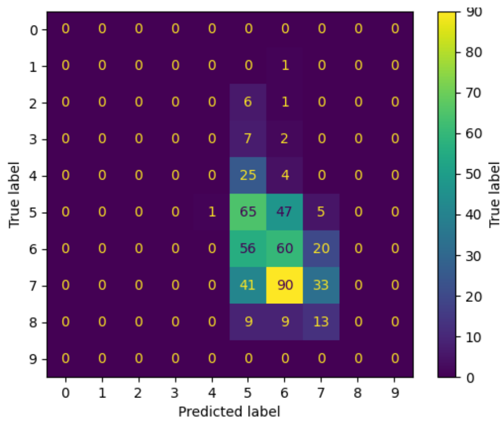
The approach of classifying ten classes is similar to the experiments in the regression section. However, instead of generating continuous values, the models predict a discrete value between one and ten. This method's possible limitation is that most readiness values are between a narrow range of values and, therefore, not evenly spread among all or most classes. This means there will be fewer data points at both tail ends of readiness, potentially leading the models to forecast most values as somewhere close to the mean.



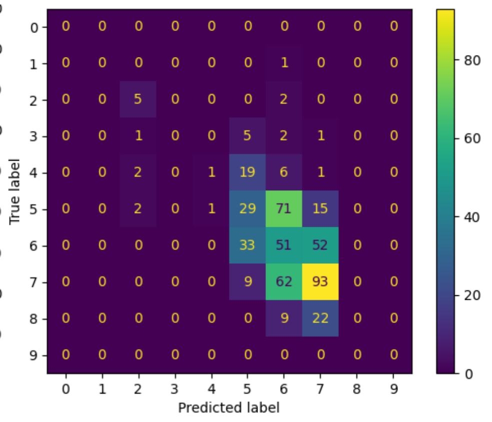
(a) XGBoost



(b) LSTM



(c) Ridge



(d) Rocket

Figure 5.10: In this figure we have four confusion matrices each produced from a unique model where we classify readiness with it's original 10 classes. We train on the whole team and forecast on the data of a single player.

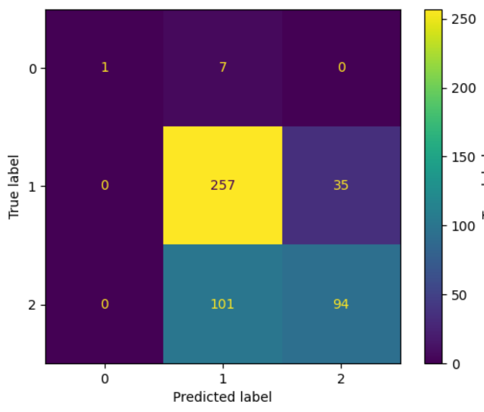
	ridge	rocket	xgboost	lstm	dummy
Accuracy	0.32	0.36	0.42	0.42	0.28
F1-Score	0.28	0.34	0.40	0.39	0.12

Table 5.8: Accuracy and F1 score for classification experiments classifying original ten classes.

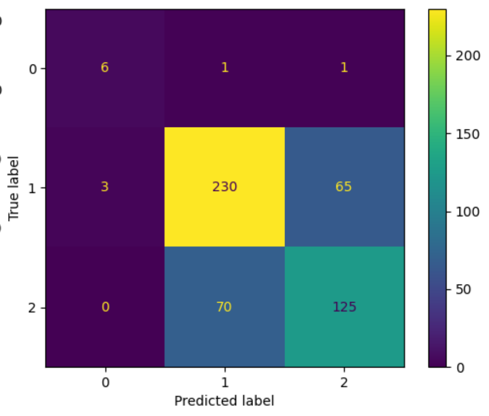
When looking at Figure 5.10 it is apparent that most of the data is largely concentrated in only a small interval of values. The accuracy and F1-scores in Table 5.8 show that XGBoost and LSTM perform the same. They have the same accuracy of 42% and only a slight difference in F1-score in favor of the XGBoost model of 40% compared to 39%. Next is Rocket with 36% accuracy and 34% F1-score. The simple Ridge classifier achieves 32% accuracy and a 28% F1-score. The Ridge classifier performs far worse than the other models and generally struggles to differentiate the classes containing the most data points. The dummy model achieves 28% accuracy and 12% F1-score. The F1-score among the best-performing models is slightly lower than their accuracy scores, indicating that the imbalanced dataset does not affect the predictions to a high degree. Generally, the accuracy and f1-score for each model are low. The models struggle to differentiate between classes, especially those close in value. Therefore, we try to simplify the problem for the next two experiments by sacrificing detailed results for simple, intuitive ones.

Forecasting Peaks

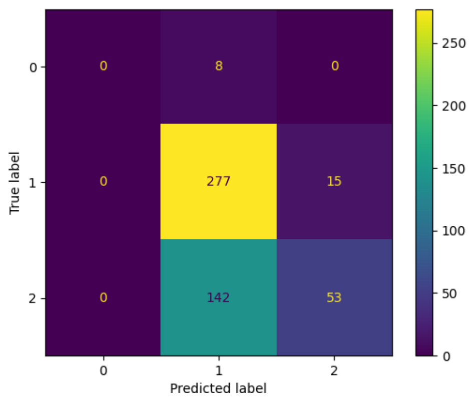
Classifying readiness with ten classes might be unnecessary, so we reduce the number of classes as seen in Figure 5.11. Instead of having ten classes, it is now three, with one defining low readiness, two defining adequate readiness, and three defining high degrees of readiness. The purpose of this use-case is to forecast peaks.



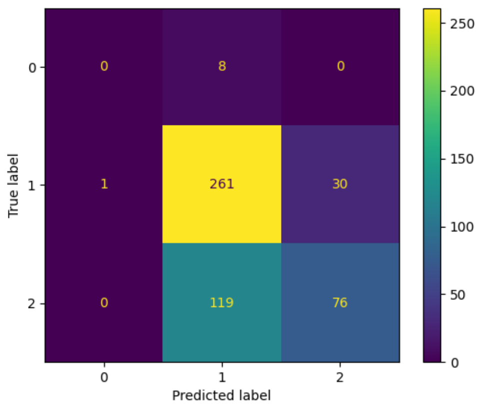
(a) XGBoost



(b) LSTM



(c) Ridge



(d) Rocket

Figure 5.11: In this figure we have four confusion matrices each produced from a unique model where we classify readiness peaks. By this we group and transform readiness into three classes. First class are all readiness values four and below, second class consists of readiness values of five to seven, and third class represents readiness values of eight and above.

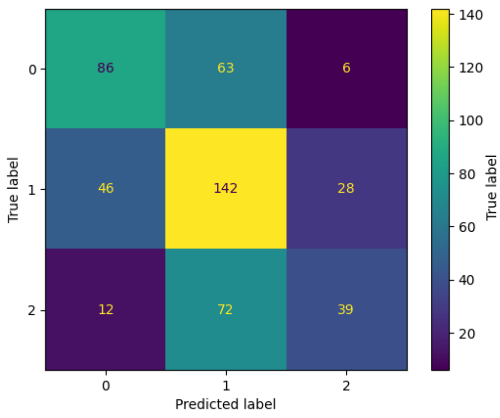
	ridge	rocket	xgboost	lstm	dummy
Accuracy	0.67	0.68	0.71	0.72	0.59
F1-Score	0.61	0.65	0.69	0.72	0.44

Table 5.9: Accuracy and F1 score for classification experiments classifying peaks.

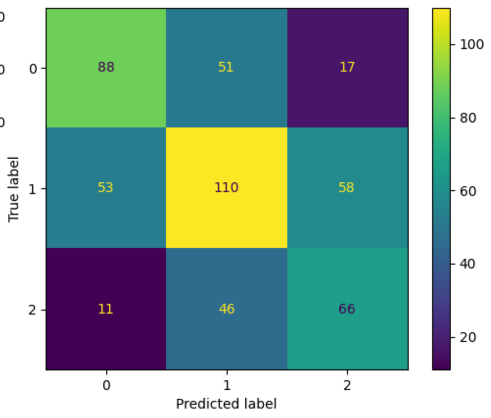
As shown from Figure 5.11 for all four confusion matrices, most data points belong to the neutral class. In this scenario, the models provide a much higher accuracy score than predicting all ten readiness classes. We observe accuracy and F1-scores in Table 5.9. The LSTM provides the highest accuracy and F1-score of 72%, followed by XGBoost 71% and 69%, respectively. ROCKET achieves an accuracy of 67% and an F1-score of 65%. The Ridge classifier obtains a 67% accuracy but a much lower F1-score of 61%. It is apparent from the confusion matrix that the Ridge classifier mostly classifies all classes as neutral. It cannot correctly classify any of the values for class zero and has the fewest true positives for class two. The baseline dummy model only predicting the majority class has an accuracy of 59% and an F1-score of 44%. The LSTM correctly classifies six out of eight peaks for class zero, which is far superior to the other models that, at most, manage one. In this scenario, the LSTM is the best-performing model with its ability to capture rare events. This approach of predicting peaks results in a skewed data distribution since peaks, especially those signaling low readiness, are rare. We will therefore look at an approach that provides a metric describing whether a player will perceive readiness for the next day better, worse, or the same as the last day.

Forecasting Positive, Negative, or Neutral Change in Readiness

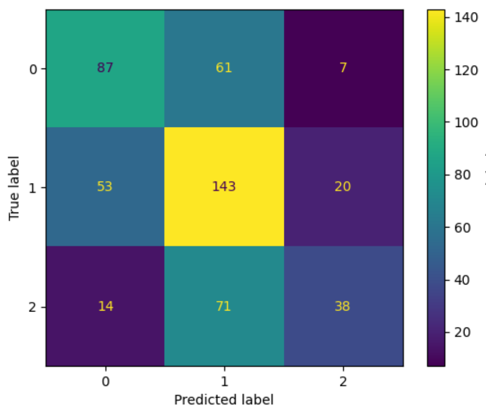
Rather than focusing on the exact value change between time steps, we now try to classify if the readiness change for the next time step is a positive, negative, or neutral development. The benefit of such an approach is that it provides an interpretable and easy-to-act-on metric. The readiness value of six might be a different standard than that of another player who reports the same. Therefore, comparing players' actual performance to an indication that they will perform worse, better, or the same as last time might be a better alternative to comparing it to an arbitrary readiness score between one and ten or one and three.



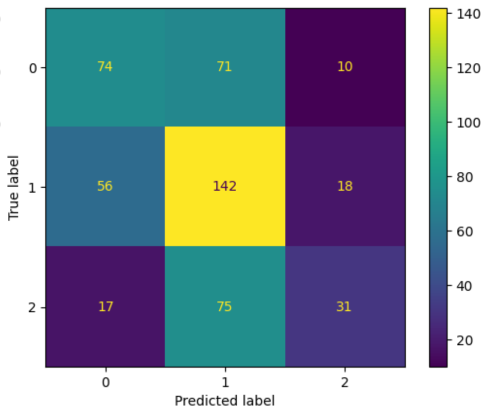
(a) XGBoost



(b) LSTM



(c) Ridge



(d) Rocket

Figure 5.12: In this figure we have four confusion matrices each produced from a unique model where we forecast positive, negative or neutral change in readiness from last to next day. We train on the whole team and forecast on the data of a single player.

	ridge	**rocket**	**xgboost**	**lstm**	**dummy**
Accuracy	0.54	0.50	0.54	0.53	0.44
F1-Score	0.53	0.49	0.53	0.53	0.27

Table 5.10: Accuracy and F1 score for classification experiments classifying readiness change.

In Figure 5.12, we see four confusion matrices expressing the results of trying to forecast these three classes. Table 5.10 shows the accuracy and F1-score obtained by the different models. We observe that XGBoost, Ridge, and LSTM have the same F1-score of 53% and accuracy of 54% with the LSTM only deviating one percentage point and with an accuracy of 53%. ROCKET has the lowest accuracy of 50% and an F1-score of 49%. The dummy accuracy of predicting only the majority class has accuracy and

F1-score of 44% and 27%, respectively. Since the most important aspect is to be able to predict when a player has a change in readiness, we observe in the confusion matrix that the LSTM has the most true positives for classes zero and two. However, it has the fewest true positives for class one.

5.3 Shortcomings and How We Can Improve Them

Our forecasts perform much better than a baseline only predicting the average. The difference in RMSE between the dummy model and the "real" models is amplified when the variance in the dataset is increased, suggesting that the models are able to learn patterns in the data. These findings are promising, but during the implementation and gathering of results, we saw possible shortcomings in our work and ideas for fixing them and improving our pipeline to produce more informative data. Therefore, the following changes justify a new chapter creating a second iteration for our implementation.

- **Time series interval:** The most notable change for the second iteration is the move from a day-to-day time series perspective to a session-to-session time series perspective. Meaning instead of the dataset consisting of all days the players have been on the team, the data we now use are only the days in which the players reported a training session or a match. This approach mitigates two issues. Firstly, by only using time steps where the players reported a training session or match, we no longer need to perform any imputation. Avoiding imputation should have a positive effect on model error since introducing more variance to the data increases RMSE. Secondly, only focusing on the actual sessions is a more true representation of the players' performances. Since the number of sessions a given player participates in differs widely, having data and models that consider this helps to tailor the overall system to each player.
- **Forecasting horizon:** As observed in this chapter, increasing the forecasting horizon beyond one day substantially increased the model error. Therefore, for the second iteration, we intend to focus mainly on the next session rather than doing large forecasting horizons. This choice is further justified by the teams with which this research initiative collaborates are more interested in how the players will perform for a given match or session than on an arbitrary day.
- **More relevant features:** Switching over to a session-based time series interval also introduces the need for new variables better describing the session and the relationship between past and future sessions. In light of this, we introduce four new variables:
  - *Session duration* is the number of minutes a session lasted for.
  - *Metabolic power* is a metric that approximates the energy used for acceleration and deceleration using GPS data.

  - *Days since last session* is the number of days between last and current session
  - *Match* is a Boolean indicating that the session is a match if '1' and a training session if '0'.

- **Objective versus subjective data:** The idea of the first iteration was to map the general well-being of the players for each day, while with the new approach, we focus more on how they will perform in a given session. Even though GPS data was less correlated to readiness than subjective data, we saw through the results of the SHAP-value plots that they have more of an impact on the actual predictions. These findings suggest that objective data describes the athletes better than the subjective quantification of things like sleep quality and mood.

5.4 Chapter Summary

This chapter presented the results of running our experiments using our pipeline. These results are meant to provide key factors important to predict readiness to train among elite female soccer players. The results can be summarized as follows:

- We observed that most models' optimal **input window size** is between five and eight, supporting a seven-day periodicity.
- We attempted forecasting for larger **output windows** but found that a forecasting horizon of more than one with our proposed methods generates too large an error.
- There was no real difference between using either forecasting method: **direct** or **one-step ahead**.
- In most cases, using a **multivariate** approach over a **univariate** one had slight benefits in reducing error. However, this depended on the model type and the specific players.
- **Team A and B** generally had the same behavior other than that team B achieved less error due to the data having less variance and examples of peak values.
- For the **classification experiments** we tested several use-cases. Peak detection and change in readiness showed the most promise in generating actionable data and accuracy. Still, the accuracy scores are low, although much higher than the dummy accuracy.

In the next chapter, we will build on what we have learned with this first iteration of our pipeline to implement potential improvements.

Chapter 6

Experiments and Results - Second Iteration

The work in chapter 5 provided the idea of a second iteration by combining the work from the first iteration with a slightly different approach described in section 5.3. In this chapter, we will reuse most of the experiments but substantially alter the time series data to accommodate a session to session based time series perspective.

6.1 Data Alterations

The data is the main change to the implementation of the first iteration to create the second iteration. We no longer perform imputation to account for missing time steps. By avoiding imputation, we observe that the number of sessions a given player has in a window of two weeks can be everything from zero to fourteen. The irregularities of when players report a session also support the idea of looking at a session-to-session perspective since the data will represent the athletes more accurately. However, a problem with this approach is the lack of data. For team A, we go from around 10 000 data points with imputation to about 4000 data points with our second iteration. This substantial decrease in the number of samples might render more data-hungry models like the TFT unusable. We have also added new variables that we explained in section 5.3. These new variables further help in explaining the general load of each session.

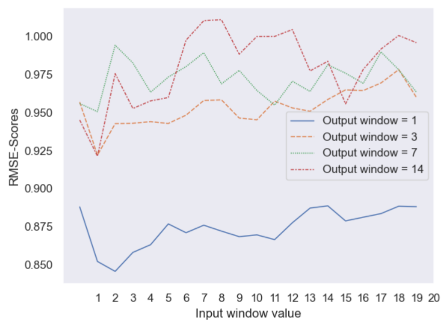
6.2 Experiments Overview

In the following experiments in this chapter, we will be reusing the same model parameters found in chapter 5 section 5.1.1. We saw no improvements using other combinations of hyperparameters. When the forecasting horizon is larger than one, we will use one-step-ahead forecasting for the relevant models. However, as mentioned in section 5.1.4, neither method has an advantage over the other. Further, except where we compare teams A and B, we only use the data from team A.

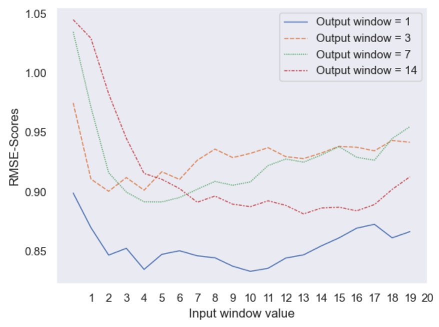
6.3 Regression Models, Second Iteration

6.3.1 Size of Input and Output Windows

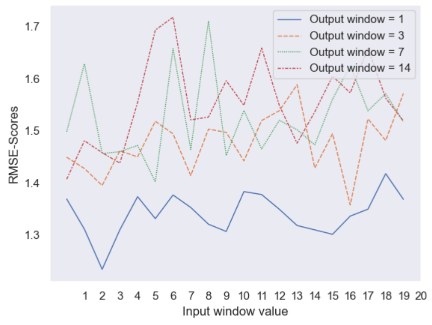
We have previously explored the concept of approximating the optimal input window size for our data. We found a considerable reduction in RMSE when increasing the size to seven from one. We also learned from the last chapter that forecasting more than one unknown time step is highly unreliable. Therefore, it makes more sense to primarily look at the optimal input window when the output window is one. We reuse the same player; in this iteration, the player consists of 220 data points. In Figure 6.1, we have generated the same plots and observe that the optimal input window for XGBoost, LSTM, and the decision tree is three. The optimal input window for the TFT model is nine, while the linear regression model achieves the lowest RMSE value when the input window is five. The benefit of choosing the optimal input window reduces model error between six and fifteen percent depending on the model.



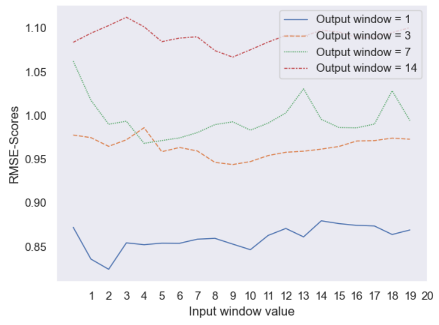
(a) XGBoost



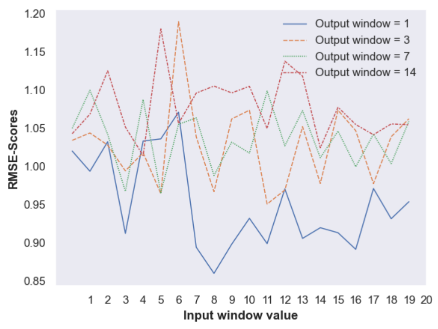
(b) Linear Regression



(c) Decision Tree



(d) LSTM



(e) TFT

Figure 6.1: Lineplots showing how different input window values affect RMSE-Scores for different types of output windows. The x-ticks denotes added size to the original output window of one. This means that the output window is two when x-tick is one.

In the second iteration, the optimal input window size is smaller, meaning less data for predictions are needed. Needing less data for predictions is generally good since it means a lesser computational cost. Since there are missing dates between sessions, the same time series dependence might be intact, as seen in the first iteration of around seven days. This is because the player used for this experiment has an average of 3.1 sessions in a week. Further, we will use an input window size of five for the remainder of our experiments in this chapter.

6.3.2 Forecasting Horizons

We produce the same experiment as in chapter 5, where we observe the change in RMSE by increasing the size of the forecasting horizon. We create the model outputs by training on team A and forecasting a player left out of the train set. The player is the same one used in Section 6.3.1

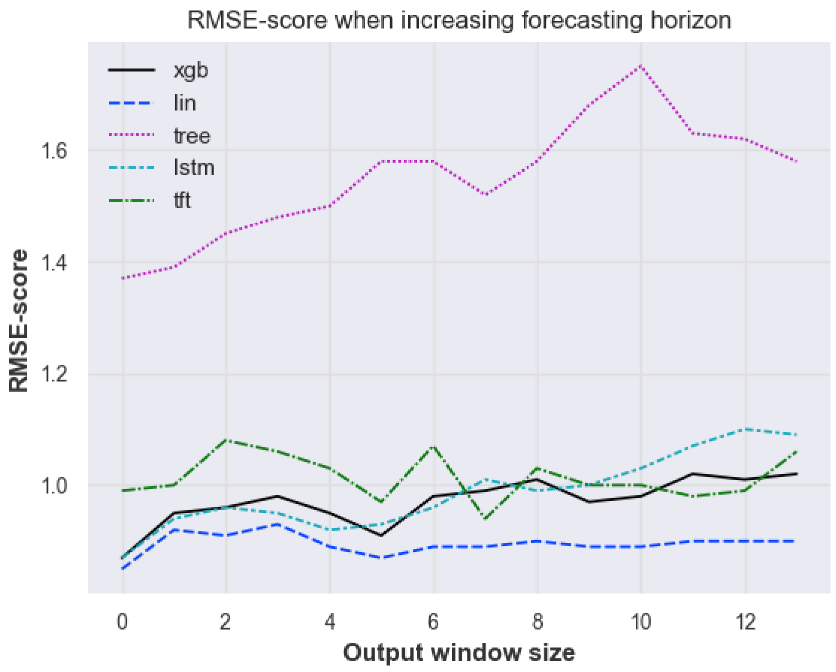


Figure 6.2: The Line-plot shows how the RMSE-scores of all the models is affected by increasing the forecasting horizon.

We observe from the plots in Figure 6.2 much of the same behavior as in the previous experiment regarding input windows in Section 5.1, but with a few distinctions. The TFT is unaffected by the output window and performs as well with low values as high. The TFT model has generally performed worse in this iteration using multivariate data as it struggles to learn relationships among features because of a lack of data. Further, the three best models, XGBoost, LSTM, and Linear Regression, all have the expected outcome of larger error with an increase in forecasting horizon. However, the increase in RMSE is lower in this iteration. It is especially noticeable when observing the linear regression model in Figure 6.2, where it trends downwards after adding three to the initial output window. This can be explained by the dataset having less variance. Therefore, sessions at random time steps are more likely to be similar. Larger output windows still result in too uncertain predictions meaning models with larger output windows produce errors similar to only predicting the mean.

6.3.3 Multivariate Versus Univariate Data

We observe how the second iteration performs using both multivariate and univariate data. In Figure 6.3, we have the same type of boxplots seen in Section 5.1.5, plotting the performance of each model and providing RMSE scores describing the overall performance of the forecasts and how the error varies depending on the player. Generally, the error compared to the previous iteration is much lower for all models except for the TFT. We attribute this to the TFT being a very complex deep learning model needing massive data to run optimally, which we have less of for this iteration.

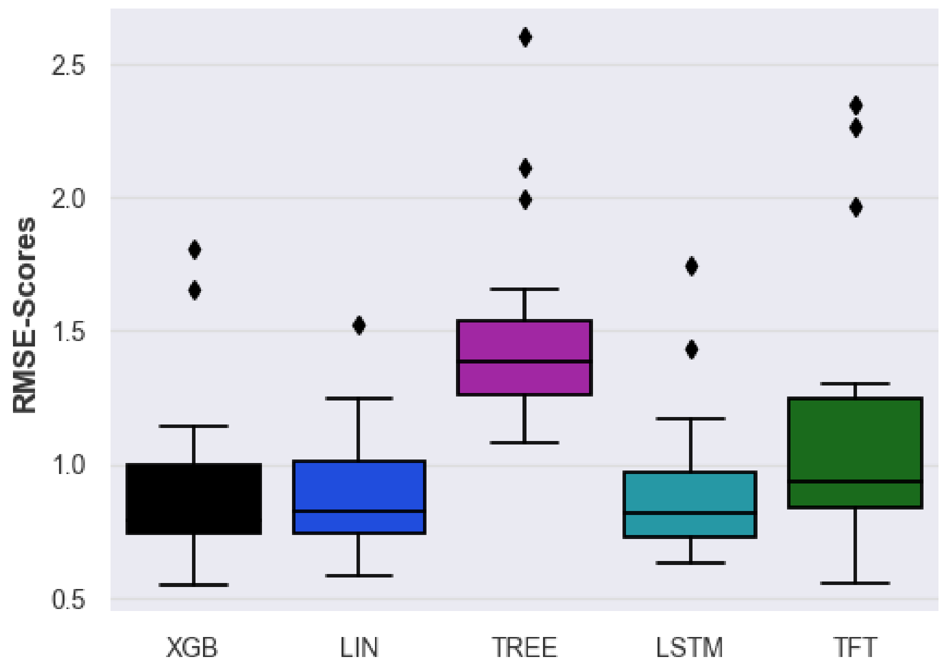


Figure 6.3: The illustration shows RMSE-Scores expressed through boxplots for all regression models using multivaraite data.

Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	0.9	0.88	1.47	0.89	1.09	1.16
Median	0.79	0.82	1.38	0.82	0.94	1.07
SD	0.3	0.22	0.33	0.25	0.46	0.42
Min	0.55	0.58	1.08	0.63	0.55	0.6
Max	1.81	1.52	2.6	1.74	2.35	2.3

Table 6.1: The table describes the boxplots in Figure 6.3 with numerical values. We also included the values from a dummy model only predicting the mean.

The decrease in RMSE in this second iteration is not necessarily a result of models performing better on the current data. Rather, the data now contains less variance and fewer extreme values, making it easier for the models to learn the data. Suppose we compare the dummy model in Table 6.2 to the dummy model in Figure 6.3. We observe for the second iteration that the median and average RMSE values are 5% and 11.8% less, respectively. These results mean less error in adopting a strategy predicting values closer to the mean in the second iteration. However, since the data in this iteration is not afflicted by imputation, it is a more true representation of the actual well-being of the athletes. Further, despite the improved results, to a certain degree being a consequence of the data being easier for the models to learn, it is still a more beneficial approach since the error is decreased and the data represents the players better.

Comparing a univariate to a multivariate approach, we observe similar results as in the previous chapter, where the error in the univariate approach is slightly higher for most models. The most apparent exception is the TFT and decision tree models performing much better using univariate data. Regarding the TFT model, this is mostly a problem attributed to the limited number of data samples. In the second iteration, we have roughly 4000 data points for team A compared to more than 10 000 in the first iteration.

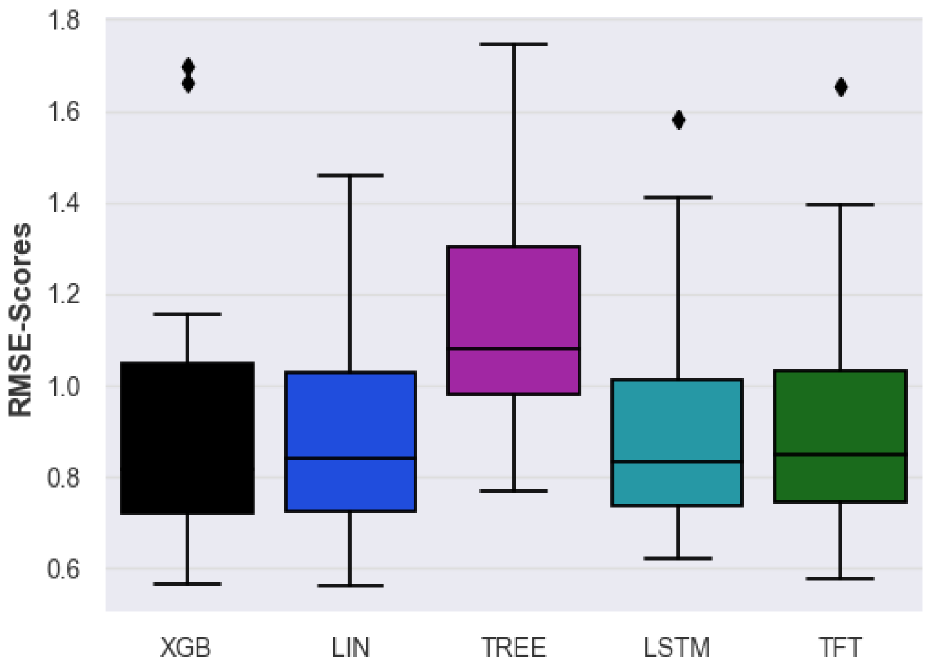


Figure 6.4: The illustration shows RMSE-Scores expressed through box-plots for all regression models using univariate data.

Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	0.92	0.89	1.16	0.9	0.91	1.16
Median	0.82	0.84	1.08	0.83	0.85	1.07
SD	0.29	0.23	0.26	0.24	0.25	0.42
Min	0.56	0.56	0.77	0.62	0.58	0.6
Max	1.7	1.46	1.75	1.58	1.66	2.3

Table 6.2: The table describes the boxplots in Figure 6.4 with numerical values. We also included the values from a dummy model only predicting the mean.

Figure 6.3 describes the percentage decrease in RMSE using multivariate over univariate data for each player on team A. The decision tree and TFT models favor univariate data for almost all players. In contrast, the models XGBoost and LSTM favor multivariate data for most players, with the Linear Regression model on average being indifferent. The LSTM at most obtains a model error decrease of 28% for one player using multivariate data but for another sees an increase in error of up to 15% using multivariate data. To summarize Figure 6.3, it shows that the benefit of using multivariate data depends on the specific player and model.

	XGBoost	Linear Regression	Decision Tree	LSTM	TFT
Player1	4.5%	7.9%	-7.5%	6.0%	1.7%
Player2	-3.6%	-6.2%	-30.0%	2.9%	-5.1%
Player3	0.1%	-13.7%	-35.0%	-8.6%	-41.3%
Player4	5.9%	8.1%	-25.0%	-13.4%	-51.3%
Player5	-2.9%	0.4%	-15.3%	-4.5%	-20.4%
Player6	2.7%	3.6%	-6.2%	11.0%	-5.2%
Player7	2.2%	-4.4%	-27.1%	-8.8%	-19.1%
Player8	0.3%	1.0%	-27.4%	0.8%	-5.9%
Player9	2.6%	-0.3%	-26.3%	4.7%	-48.8%
Player10	7.4%	9.6%	-4.3%	16.4%	-8.8%
Player11	-5.8%	-4.9%	-21.7%	-3.3%	-21.8%
Player12	3.3%	0.8%	-7.6%	3.8%	-4.9%
Player13	-0.6%	-7.4%	-33.0%	-8.7%	5.9%
Player14	11.1%	13.9%	-3.3%	16.2%	-14.4%
Player15	3.9%	4.5%	-14.6%	8.7%	-6.3%
Player16	0.3%	-1.8%	-22.0%	-0.9%	-16.4%
Player17	6.1%	-0.2%	-38.1%	-3.3%	-13.2%
Player18	0.4%	-1.6%	-36.2%	-0.3%	-23.5%
Player19	1.5%	4.5%	-18.1%	-7.8%	-25.5%
Player20	7.1%	8.5%	-12.5%	3.0%	-1.2%
Player21	-7.4%	-8.3%	-29.7%	-15.3%	-36.5%
Player22	1.6%	1.9%	-24.5%	3.1%	-14.3%
Player23	-2.9%	1.0%	-30.0%	0.2%	3.8%
Player24	5.1%	0.1%	-26.2%	-10.5%	-33.3%
Player25	8.1%	10.8%	-6.9%	28.3%	-2.7%

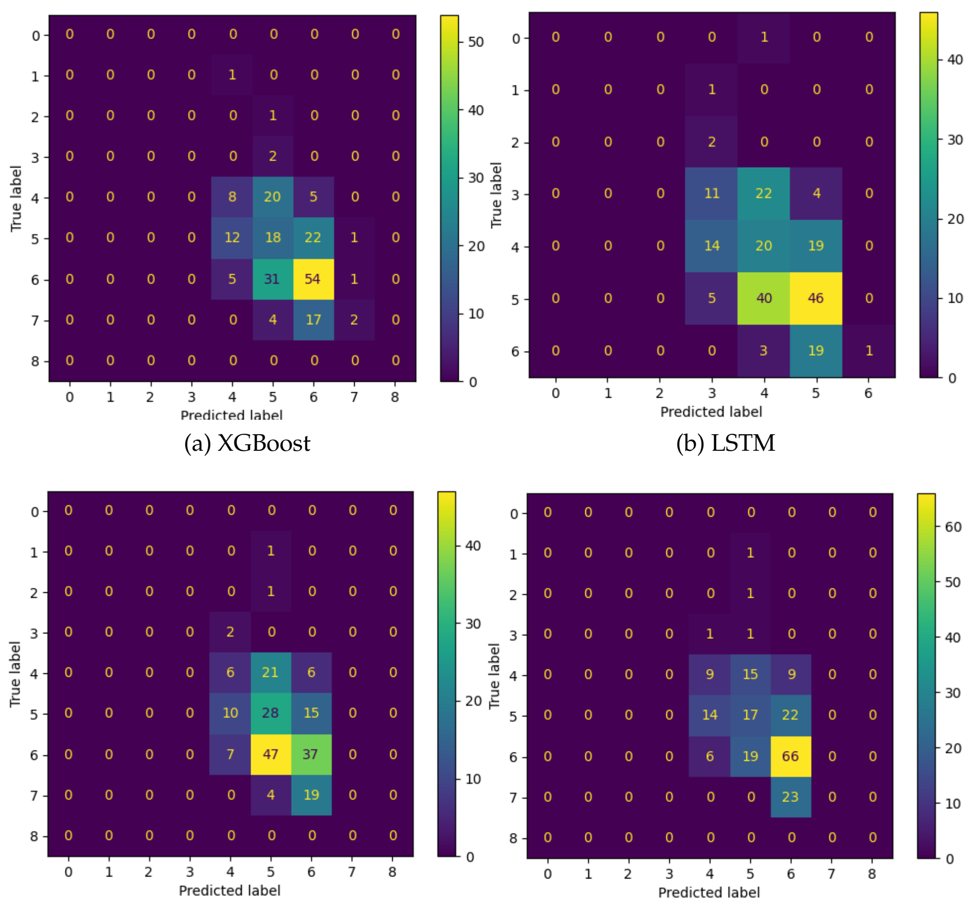
Table 6.3: Table showing percentage decrease in loss using multivariate over univaraite data for all players on team A for all models. Positive values indicating better results using multivariate data.

6.4 Classification Models, Second Iteration

The same classification experiments conducted in chapter 5.2 are also recreated for this iteration using our altered time series dataset. We reuse the same model hyperparameters. We apply an input window of five and an output window of one. We also reuse the same player, although the player now has fewer data points than in the last iteration. Further, this section will go through these experiments.

Forecasting The Original 10 Classes

We use the original ten classes in this experiment and observe the results in Figure 6.5 and Table 6.4. ROCKET has by far the highest accuracy of 45% but an F1-score of 41%. Xgboost achieves an accuracy of 40% and an F1-score of 39%, followed by the LSTM obtaining an accuracy of 38% and an F1-score of 36%. Aside from the dummy model, the Ridge classifier performs the worst with 35% accuracy and 32% F1-score. In this iteration, we see no substantial improvement in accuracy and F1-scores.



(a) XGBoost (b) LSTM

(c) Ridge (d) Rocket

Figure 6.5: In this figure we have four confusion matrices each produced from a unique model where we classify readiness with it's original 10 classes. We train on the whole team and forecast on the data of a single player.

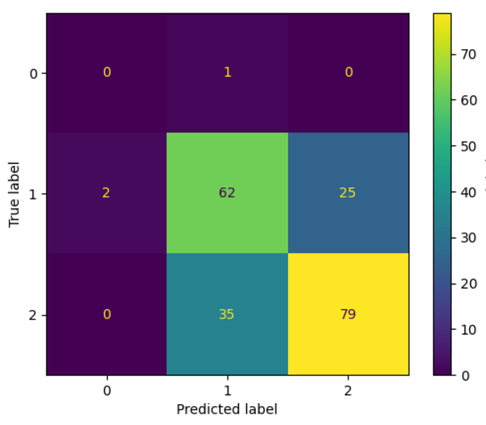
	ridge	**rocket**	**xgboost**	**lstm**	**dummy**
Accuracy	0.35	0.45	0.40	0.38	0.26
F1-Score	0.32	0.41	0.39	0.36	0.11

Table 6.4: Accuracy and F1 score for classification experiments classifying original ten classes.

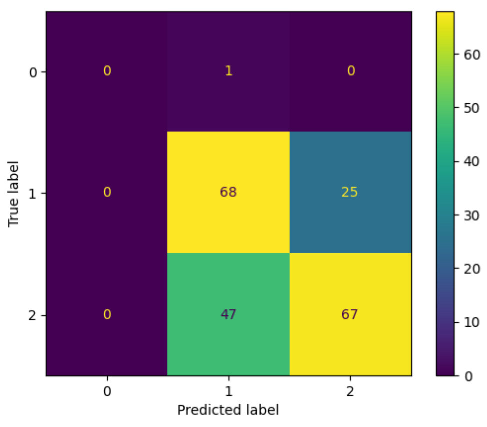
As discussed in Chapter 5, this use-case struggles to achieve high accuracy scores largely because the models struggle to differentiate between similar readiness values. However, the next classification experiments rectify this issue and simplify the approach of predicting readiness to provide metrics capturing important events.

Forecasting Peaks

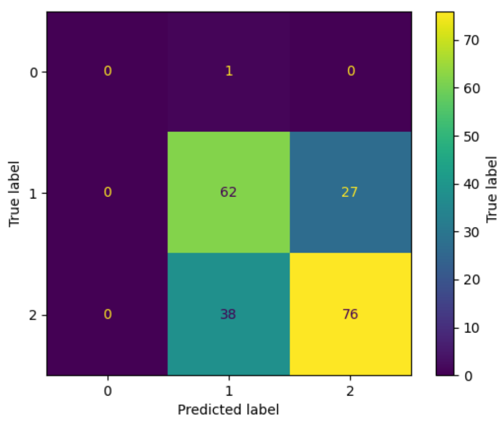
The idea of forecasting peaks shows promise and achieves the highest accuracy among our classification experiments. We observe the results in Figure 6.6 and Table 6.5. The XGBoost model achieves the highest accuracy and F1-score of 69%, followed by the Ridge classifier with 68%. The LSTM achieves 65% for both accuracy and F1-score, while this number is 63% for the ROCKET model. The accuracy and F1-score for the dummy model are 44% and 27%, respectively. In this case, there is only one value for class zero which none of the models are able to classify correctly.



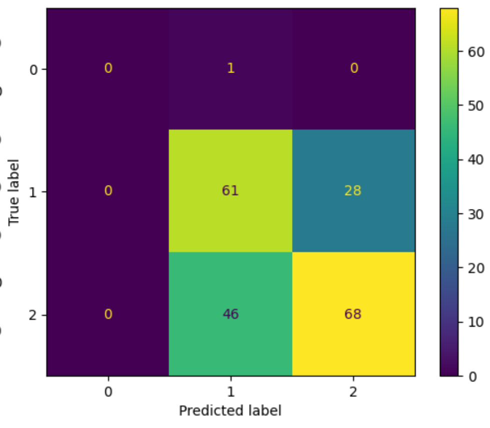
(a) XGBoost



(b) LSTM



(c) Ridge



(d) Rocket

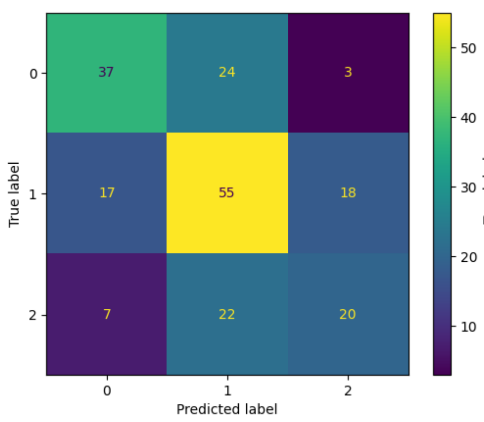
Figure 6.6: In this figure we have four confusion matrices each produced from a unique model where we classify readiness peaks. By this we group and transform readiness into three classes. First class are all readiness values four and below, second class consists of readiness values of five to seven, and third class represents readiness values of eight and above.

	ridge	**rocket**	**xgboost**	**lstm**	**dummy**
Accuracy	0.68	0.63	0.69	0.65	0.44
F1-Score	0.68	0.63	0.69	0.65	0.27

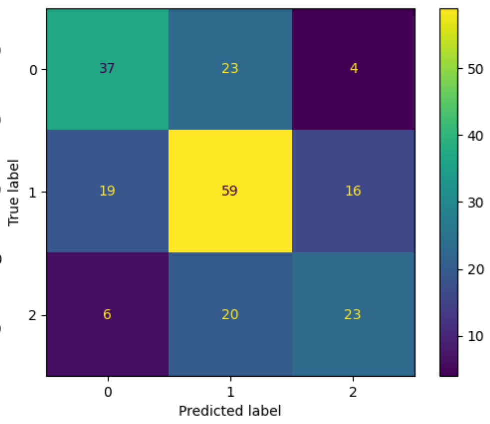
Table 6.5: Accuracy and F1 score for classification experiments classifying peaks.

Forecasting Positive, Negative, and Neutral Change in Readiness

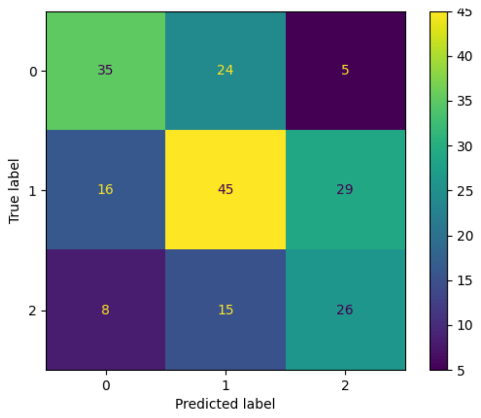
In this classification experiment, we predict positive, negative, or neutral changes in readiness compared to the last session. We observe the results in the confusion Matrices in Figure 6.7 and accuracy and F1-score in Table 6.6. We observe that the LSTM performs best with an accuracy of 58% and an F1-score of 57%. This is followed by XGBoost achieving 55% for both accuracy and F1-score. The Ridge classifier and ROCKET obtain about the same accuracy and F1-score. Our dummy model only predicting the majority class has an accuracy of 44% and an F1-score of 27%.



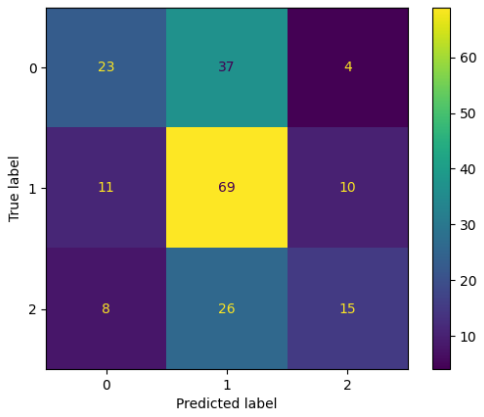
(a) XGBoost



(b) LSTM



(c) Ridge



(d) Rocket

Figure 6.7: In this figure we have four confusion matrices each produced from a unique model where we forecast positive, negative or neutral change in readiness from last to next day. We train on the whole team and forecast on the data of a single player.

	ridge	**rocket**	**xgboost**	**lstm**	**dummy**
Accuracy	0.52	0.53	0.55	0.58	0.44
F1-Score	0.52	0.51	0.55	0.57	0.27

Table 6.6: Accuracy and F1 score for classification experiments classifying readiness change.

This approach classifying change of readiness offers a unique perspective that considers the development of readiness regarding the specific players. For this method, only the perceived change of readiness among players matters, not the readiness score itself. However, an accuracy of 58% is still generally low for a real-world application.

6.5 Comparing First and Second Iteration Results

Overall, the first iteration has a much higher RMSE score than we observed in the second iteration of our pipeline. Also, for classification experiments, we saw a slight increase in accuracy in the second iteration of our pipeline, except for peak classification, which achieved three percent less accuracy and F1-score. However, these preliminary results are limited and lack conclusive evidence. In order to further verify what approach has the best potential, we need to look at the following:

- Compare dummy predictions to actual predictions across both iterations and evaluate the differences between dummy scores and actual predictions.
- Compare the RMSE of the first iteration using imputed data but only using prediction values from actual days to the RMSE of the non-imputed second iteration predictions.

6.5.1 Comparing Dummy classification Results Across Iterations

The classification results are similar for both pipeline iterations, although the second iteration has slightly better results. By comparing these results to dummy models, we can determine which approach has the greatest gap in accuracy and F1-score between dummy models and the other ML models.

First Iteration- Accuracy			
Model	Original Classes	Peaks	Change
XGBoost	42%	71%	54%
Ridge	32%	67%	54%
ROCKET	36%	68%	50%
LSTM	42%	72%	53%
Dummy	27%	59%	44%
diff- dummy and best model	55%	22%	23%
Second Iteration- Accuracy			
Model	Oirginal Classes	Peaks	Change
XGBoost	40%	69%	55%
Ridge	35%	68%	52%
ROCKET	45%	63%	53%
LSTM	38%	65%	58%
Dummy	26%	44%	44%
diff- dummy and best model	73%	57%	30%

Table 6.7: The table shows all accuracy scores across both iterations, all classification models, and dummy models.

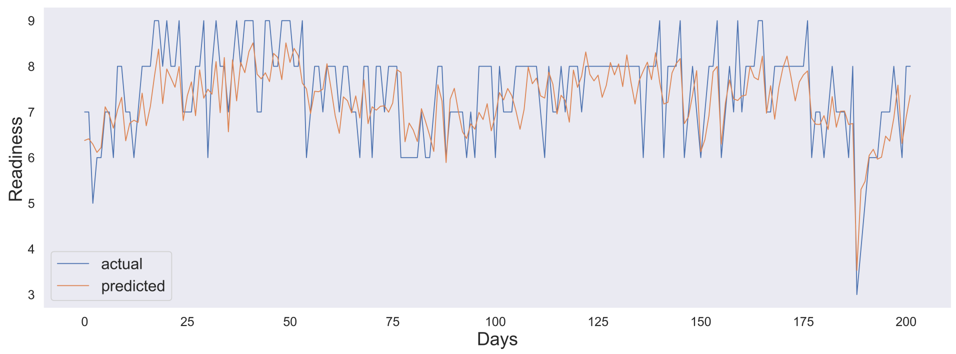
First Iteration- F1-Score			
Model	Original Classes	Peaks	Change
XGBoost	40%	69%	53%
Ridge	28%	61%	53%
ROCKET	34%	65%	49%
LSTM	39%	72%	53%
Dummy	12%	44%	27%
diff- dummy and best model	330%	64%	96%
Second Iteration- F1-Score			
Model	Oirginal Classes	Peaks	Change
XGBoost	39%	69%	55%
Ridge	32%	68%	52%
ROCKET	41%	63%	51%
LSTM	36%	65%	57%
Dummy	11%	27%	27%
diff- dummy and best model	373%	256%	204%

Table 6.8: Table showing F1-scores for all classification tasks across both iterations.

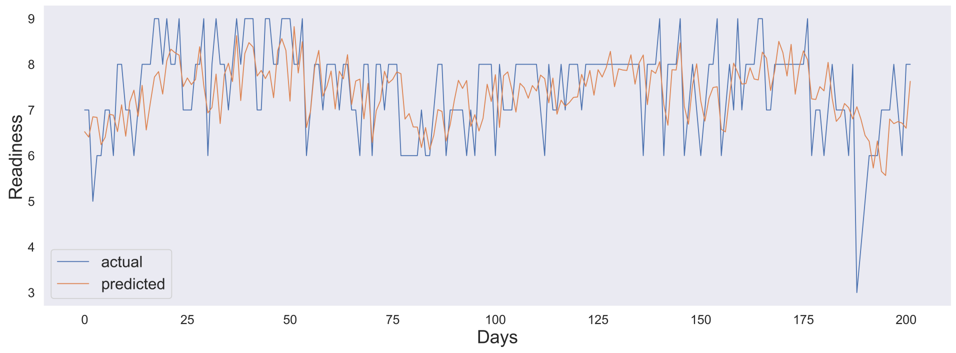
Tables 6.7 and 6.8 show all models' overall accuracy and F1-scores for all classification use-cases. The table also denotes the greatest difference in accuracy and F1-scores between the dummy model and the best-performing model for each use-case. The best solution forecasting original classes was ROCKET in the second iteration achieving accuracy and F1-score 73% and 373% higher than the dummy model. For comparisons, the best model in the first iteration predicting ten classes achieved 55% better accuracy and 330% better F1-score than the dummy model. Since the F1-scores and accuracy are almost identical for the best-performing models, only mentioning accuracy is adequate. The first iteration has slightly higher accuracy and F1-scores in the use-case predicting peaks. However, the first iteration only has 22% better accuracy than the dummy model, while the second iteration has an accuracy 57% better than the dummy model. For the third use-case predicting change in readiness, these numbers are 23% in the first iteration and 30% in the second iteration in terms of accuracy. For all use-cases, the second iteration performs much better than the dummy model compared to the results in the first iteration. Since both iterations forecast dissimilar data, knowing how they perform compared to each other and the dummy models provide a necessary context for evaluating these approaches. The second iteration achieves far better results for the use-cases original classes and change in readiness but is slightly worse regarding peak prediction. However, as stated, the second iteration performs much better than the dummy model compared to the first iteration. It then stands to reason that the second iteration might be better, although with lower accuracy. These findings suggest that the second iteration is performing better for classification tasks.

6.5.2 Comparing Imputed and Non-Imputed Predictions

In this comparison, we want to observe whether the error for the days that are not a result of imputation is lower for the forecast using imputed data compared to the forecasts only using non-imputed data. By this, we mean to filter out the predictions only predicting an imputed value, so we are left with only predictions predicting real data that the players filled in. This comparison is to observe whether imputed data improves results for session to session based predictions. Also, the imputed values might be noisy and contain more variance than the real data. Therefore, this comparison will provide better insight into the usefulness of our imputation method.



(a) Imputed



(b) Not Imputed

Figure 6.8: Image 6.10a shows the predicted versus the actual readiness values when forecasting using the dataset from the first iteration but only viewing the predictions/days that were not imputed. Image 6.10b shows the predicted versus the actual readiness values when forecasting using the dataset from the second iteration. Plot 6.10a has an RMSE of 0.74 while Plot 6.10b has an RMSE of 0.88.

Figure 6.8 displays two plots where Plot 6.10a shows predicted versus actual values using the imputed dataset, while Plot 6.10b shows predicted versus actual values using the non-imputed dataset from the second iteration. By only glancing at the plots, we see that the plot using imputed data performs much better than the one not using imputed data. The plot also manages to follow the largest peak to the downside correctly. Plot 6.10a has an RMSE of 0.74 while Plot 6.10b has an RMSE of 0.88, meaning that the method not using imputation has 19% more error. So far, this method provides the least amount of error, suggesting that imputed data is able to contextualize missing data. Therefore, the better approach is using imputed data to predict the next session, not the day. Next, we will examine how this approach performs using leave-one-out cross-validation on team A.

We recreated the same boxplots previously used in this chapter but for the approach using imputed data to only predict actual days. The boxplots, available in Figure 6.9, show a reduction in RMSE compared to the previous experiments found in Figures 5.5 and 6.4 for all models except the TFT. In table 6.9, we observe numerical values describing the boxplots in more detail. The mean and median for XGB, Linear Regression, and LSTM are very similar, and the lowest produced by our experiments. These results show that using imputation can offer better results for these types of time series forecasts.

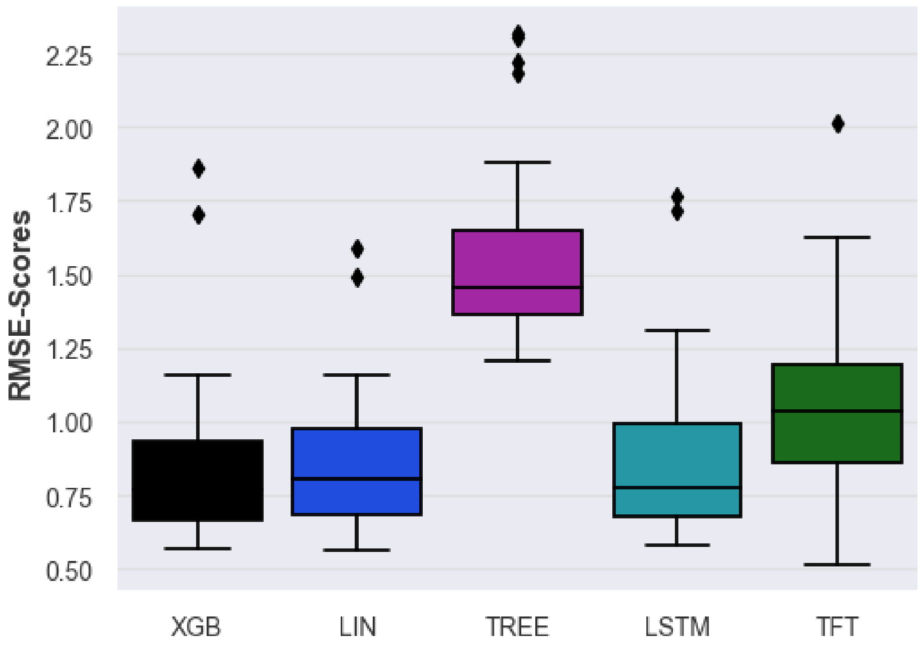
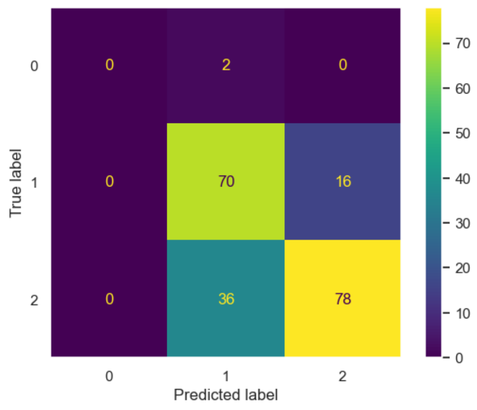


Figure 6.9: The illustration shows RMSE-Scores expressed through boxplots for all regression models. This approach uses only the predictions from the first iteration that are not imputed dates to calculate RMSE.

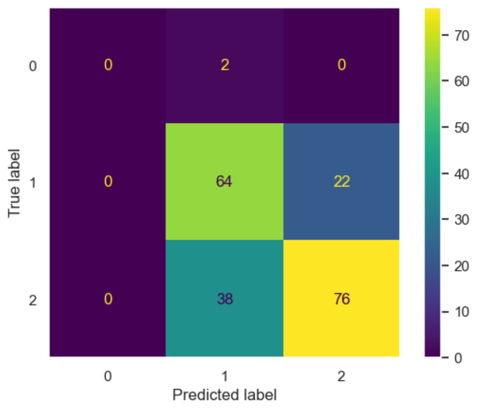
Metrics	XGB	Linear Regression	Decision Tree	LSTM	TFT	Dummy
Mean	0.86	0.85	1.59	0.88	1.08	1.16
Median	0.77	0.81	1.46	0.77	1.04	1.07
SD	0.32	0.26	0.34	0.31	0.32	0.42
Min	0.57	0.56	1.21	0.58	0.52	0.6
Max	1.87	1.6	2.32	1.77	2.02	2.3

Table 6.9: The table describes the boxplots in Figure 6.9 with numerical values. We also included the values from a dummy model only predicting the mean.



	0	1	2
0	0	2	0
1	0	70	16
2	0	36	78

(a) Uisng imputed data



	0	1	2
0	0	2	0
1	0	64	22
2	0	38	76

(b) Not using imputed data

Figure 6.10: This figure shows two confusion matrices with one using imputed data to generate predictions while the other one only uses actual days. Figure 6.10a has accuracy and F1-score of 73% while Figure 6.10b has accuracy and F1-score of 69%.

Figure 6.10 shows two confusion matrices predicting peaks. Figure 6.10a uses imputed days to predict actual days, while Figure 6.10b only uses actual days to predict actual days. We use XGBoost since it provided the best result for this use-case. The approach using imputed data achieves accuracy and F1-score of 73%, while the other approach only using actual days achieves accuracy and F1-score of 69%. Also, for classification tasks, using imputed data to predict actual days improves overall accuracy and F1-scores.

6.6 Chapter Summary

The objective of this chapter was to apply the improvements we derived from creating the first iteration of our experiments to the next iteration of experiments. We discovered that a session-to-session time series perspective using non-imputed data (except for the TFT) preferred a smaller input window between three and five. The overall forecasting error when increasing the output window size decreased, most likely due to lower variance in the dataset, making the strategy of predicting the mean more viable. However, this did not change the fact that a forecasting horizon equal to one has a much lower RMSE than a forecasting horizon above one. In most cases, multivariate data was beneficial but less so compared to the first iteration. Still, this is very dependent on the player. Some players achieve higher scores with multivariate data, while others achieve higher scores with univariate data. Choosing a feature selection strategy for each player should drastically decrease RMSE. As for classification tasks, we saw a slight increase in accuracy for the second iteration.

Further, the second iteration did much better than the dummy models compared to the first iteration. We also compared the non-imputed and imputed predictions using only the dates available in both predictions and found that using imputed data resulted in lower RMSE. Therefore, using imputed data offered the best results when only using predictions from actual days. The approach using imputed data to predict actual days improved results for both regression and classification tasks.

Chapter 7

Discussion

In this chapter, we intend to further elaborate on the results obtained from the experiments in chapter 5 and 6. We will discuss the significance of our findings and describe the general limitations of our methods and their ability to forecast athlete data. We will also mention ethical considerations relevant to the work in this thesis.

7.1 Daily and Session Based Time Series Intervals

The main motivation behind creating the second iteration was to implement a more consistent and practical approach to time series forecasting of readiness. The models predict data more true to the original domain by focusing more on a session-to-session time series perspective. As discussed earlier, teams are interested in athletes' performance during important events. A session to session based time series perspective supports this statement since we only use data from days when players attended relevant soccer activities. Capturing important events is relevant to all sports, making this method applicable to other sports.

We observed lower RMSE values using a session-to-session time series perspective in our regression experiments. The reduction in RMSE is largely because the implemented imputation method we used in the first iteration increased the variance in our dataset. In turn, making it more difficult for the models that already struggled to forecast values different from the mean. Also, since we included all days in a player's active periods, the imputation algorithm had to, in several cases, impute more than 50% of a player's days. Imputation to such an extent causes uncertainties in the actual practical use of the model outputs since a large part of the data is artificially generated. However, with our imputation method, IterativeImputer, we discovered that the first iteration was more accurate when only accounting for predictions for non-imputed dates.

7.2 Imputation

The first iteration of our pipeline utilized imputed data. This data configuration resulted in a dataset with greater variance and, therefore, higher model error than forecasting using non-imputed data. However, in Section 6.5.2, we discovered that in the first iteration, when only accounting for the predictions of actual days, the error was lower than that found in the second iteration. Suggesting that the imputed data helps give context to the models for actual days that do not result from imputation. Given that the data used in this thesis is sparse and hard to obtain, it is beneficial that imputation is a viable method capable of improving results.

7.3 Input And Output Window Sizes

In our experiments, we observed different optimal input window sizes across both experiment iterations and models. However, the deviations only applied to the worst performing models: Decision Tree and the TFT. The remaining three models, XGBoost, Linear Regression, and LSTM, had the same behavior with only a slight difference in optimal input window size. The optimal input window size for the first iteration is between five and eight, while for the second iteration, it is between three and five. Regarding the first iteration, these three models show that using around the previous week provides the lowest forecasting error. The second iteration has a variable time interval where there can be anywhere from zero to seven or more sessions in a week. The player we used for these experiments has an average of 3.1 sessions a week. In the second iteration, the optimal input window size is between three and five, indicating that the sessions from the past week provide the best results. Using the past seven days as the input window value was also discussed in related work by Kulakou et al. [34] to be optimal.

We also observed that the models performed better using the previous seven days as the input window when increasing the output window size, as shown in Figure 5.1. This means that output window size is irrelevant in determining input window size. Further, the greatest decrease in model error happened, going from input windows one to three. Therefore, only looking at the previous time step is suboptimal in understanding the athletic state of athletes, at least with our approach and selection of ML models.

Generally, we observed that increasing the forecasting horizon from one greatly increased model error. Forecasting multiple days or sessions is difficult or impossible with the data and methods proposed in this thesis. Since readiness is a measure of perceived ability to perform, other factors that the models have no knowledge of might impact readiness. Additionally, after increasing the forecasting horizon to three days, the model error did not substantially change when the forecasting horizon further increased. Therefore, forecasting three time steps into the future was as accurate as forecasting two weeks into the future. With larger forecasting horizons, we see that the models are predicting values close to the mean.

7.4 Multivariate Versus Univariate Data

On average, using multivariate data reduces the overall error of the model forecasts. However, this difference was small. By mapping individuals, we observed a high discrepancy between players and whether or not using multivariate data has a positive outcome on predictions 6.3. For some player's data, when looking at the best performing models, the reduction in RMSE was as high as 28% using multivariate data; for others, the RMSE increased up to 15% compared to using univariate data. In our comparisons of teams A and B in Tables 5.6, we saw that team B on average, had less of a positive effect using multivariate data than team A. Team B also has less variance. We observed no significant relationship between variance and benefits in using multivariate data, as shown in the analysis in Figure 9.7. We found no common denominator indicating when a player will benefit from using multivariate data. However, we speculate that this is dependent on the subjective nature of the player and whether the data from the survey is able to represent the well-being of the athlete. Further investigation is needed to determine when certain features become relevant in predicting readiness.

In cases where the models were bottlenecked by their complexity level or lack of data, using multivariate data caused worse performance. We observed this for the decision tree and the TFT model. The decision tree struggles with multivariate data and learning the complex relationships among multiple features, while the TFT needs more data to learn these relationships. The TFT performed worse in the second iteration, where data was more sparse.

7.5 Relevant Features and Data Types

In our data analysis, we discovered how the impact of specific features on predictions depended on the player's subjective nature. For some players, GPS-derived features had a big impact; in others, it was as impactful as adding random noise. Also, using too many features in other cases resulted in worse performance than a univariate approach. However, on average multivariate data yielded better results than univariate. Based on these findings, the main issue is that each test set, representing a player, deviates from the training data in an unforeseeable way. Therefore, ranking the relevant features after the most to least impact on predictions is not obvious.

For our experiments, we chose not to use feature selection. Our reasoning is that the preferred features in the train set differ from those in the test set. Our analysis and experiments showed that player predictions responded differently to different features, suggesting that the models have issues generalizing to all players.

Since we are training on the team rather than the relevant player, the model is not learning data specific to that player. A possible method to improve this issue is to use a form of transfer learning like fine-tuning. In this case, the weights from the model trained on the team are used as a base, and the model is further trained on a percentage of the relevant player to better capture the unique patterns of the specific player's data.

7.6 Evaluation of Selected Models

In our experiments, both regression and classification, we tested six different unique models for the first and second iterations: **LSTM**, **XGBoost**, **Linear model**, **Decision Tree**, **TFT**, **ROCKET**. As previously mentioned in Section 2.5, our reasoning for this selection of models is based on criteria such as state-of-the-art results in the time series domain and model complexity.

For regression tasks, we found that the Linear Regression model, LSTM, and XGBoost performed best for both iterations and obtained, in essence, the same quantity of error. This was the case for all experiments; univariate and multivariate data and forecasting horizons. The tree model overall was highly inaccurate and is not worth mentioning. The TFT generally performed worse than the other three best models in the first iteration. For the second iteration, the TFT Performed much worse with multivariate data but had only slightly higher RMSE when using univariate data relative to the other best-performing models. We attribute the poor performance to a lack of data and an inability to ignore unimportant features.

In our classification experiments, we presented results for three different use-cases. We observed that the linear Ridge classifier obtained the lowest accuracy scores. ROCKET performed exceptionally well in one example where we predicted the original classes in the second iteration. The LSTM and XGBoost models achieved the highest accuracy in most examples. They were also in addition to the Linear Regression model the best performing models for the regression tasks. Despite linear regression performing well for different train/val/test splits and leave-one-out cross-validation, it is still highly likely that the linear regression is overfitting to the data.

7.7 Use-Cases

In our experiments, we presented three different use-cases: Prediction using all ten classes, predicting peaks, and predicting positive, negative, or neutral change in readiness. The motivation behind presenting several use-cases was to investigate how to best generate data that can aid in decision-making by players and coaches.

Regression values are the least practical approach to present data among our selection of use-cases since it consists of arbitrary continuous numbers between one and ten. Also, this approach does not consider that what some players perceive as seven might be perceived as five, among others. Therefore, the use-case does not consider the subjective nature of perceived readiness. We also saw for classification that the original ten classes at most had an accuracy of 45%, which is low. The use-case predicting peaks offers a more robust way of interpreting readiness by differentiating between important and unimportant events. High values above seven and low values below five should reflect the same perceived readiness better among all players, as these are rare occurrences. Differentiating between neutral readiness values between five and seven is not as important as capturing peaks. The final use-case considers the relative development of readiness only indicating if the next session will have a lower, higher, or the same readiness score. How well a player perceives their abilities are subjective, and a use-case that accommodates the subjective nature of players in a simple and interpretable way is the better approach. Therefore, the previous two use-cases show the most promise for a real-world application.

7.8 Hyperparameter Tuning

For our models, we used either the hyperopt [5] library for hyperparameter tuning, manual tweaking, or author recommendations for hyperparameters. Further research into hyperparameter tweaking is possible to achieve even greater results.

For the LSTM model, we saw a benefit in using two LSTM layers. However, this did not benefit the TFT model. Dropout is a regularization technique that prevents models from overfitting by avoiding co-dependence between units. We tested different levels of dropout for both deep learning models and observed no positive effect on the LSTM and only a slight improvement for the TFT. An increase in batch size above 32 led to worse results and longer training because more epochs were needed despite the faster GPU performance. For the XGBoost, we observed that the model was prone to overfitting when using the hyperopt optimization tool with too wide parameter range. Especially the parameter max depth would lead to overfitting of the model if left too high. Therefore, we only searched max depth values between 1 and 12 for our hyperparameter tuning. Also, XGBoost offers regularization which we enabled.

7.9 Ethical Considerations

It is important to reflect upon the consequences of research and the potential misuse and harm it can cause. Therefore, it is crucial to consider how the given research impacts systems and, subsequently, people, as Saltz et al. [51] argued in their paper which discusses the ethical implications of research in data science. Following is a discussion of possible ethical issues relevant to this thesis.

A tool created to show the future performances of athletes can cause unfair situations where some players are consistently more favored than others. In other words, our model is biased towards some or a combination of data. These types of biases are one of the key issues in data science addressed by Saltz et al. [51]. In the case of this thesis, all data are either health or soccer performance related and are supposed to provide important insight into every player involved. Therefore, it is crucial that models work properly, or it could cause harm to the overall success and well-being of the team/player. Therefore, it is important to analyze the features and use feature importance metrics in a real-world application to understand how the models weigh specific predictions.

Using readiness scores to actively choose players for important events might cause issues if players inaccurately report high readiness values to increase the likelihood of being chosen. System-wise, it would result in unreliable predictions since the data is untrue. From a wellness perspective, this could lead to players fixating on an arbitrary number and induce stress which is counter-intuitive to the overall goal of enhancing training conditions and game strategies. Therefore, the overall system needs to be used ethically and with consideration for the players' well-being. Incorporating a trustworthy AI approach where each part can be explained and accounted for should be a priority, especially when predictions directly impact people.

A problem that has plagued research communities for years is p-hacking [24]. In short, it means producing unrealistic results through selective data selection and analysis. These methods turns insignificant results significant [24]. Such practices result in misleading results and are often impossible to replicate [24]. Therefore, the work in this thesis is reproducible. Further, our code is also available through GitHub.

7.10 Chapter Summary

In this chapter, we discussed the significance of our key findings and how they overall relate to our research question to determine a good approach to predict readiness. We also described relevant ethical considerations.

Chapter 8

Conclusion

In this thesis, we present several data and model configurations and three use-cases to determine what factors have the greatest impact in forecasting readiness to train among elite female soccer players to provide actionable data. Our work is structured as a four-step pipeline to efficiently and rigorously generate results from our experiments. These steps include data importing, data analysis and pre-processing, experiments, and evaluation.

We chose a novel approach by combining subjective wellness and GPS data with several state-of-the-art ML models for time series forecasting. We leverage complex imputation to contextualize the missing data. We also thoroughly analyze the dataset, both with and without imputed data. The experiments in our pipeline determine what impact several data and model configurations have on forecasts, such as optimal input window size, forecasting horizon, forecasting method, univariate and multivariate data, and different use-cases using regression and classification. We make model comparisons using these configurations to derive results. The structure of our pipeline makes for easy implementation of new experiments compatible with our processed data and plots for visualization.

Our experiments show several key factors important when forecasting readiness: By providing context to missing data in the form of complex imputation, we see a reduction in model error. Further, there is a great discrepancy between what is important to each unique player. For some players, using more features had the same result as adding random noise, while it greatly increased accuracy for others. Therefore, dynamically choosing the optimal data and model configuration for each unique player will greatly impact accuracy. Regarding the arbitrary nature of readiness, use-cases that define important moments, such as peak detection, or the relative change in performance, such as predicting the type of change, is a more interpretable metric providing more reliable data to make decisions.

8.1 Revisiting the Problem Statement

From Section 1.2, we derived five sub-questions to answer our main research question adequately.

- **Q1: Is the use of complex data imputation useful to our forecasts?** We discovered that the use of complex imputation, namely IterativeImputer, could provide reasonable context to the missing data to improve results.

- **Question 2: What number of prior time steps is optimal to use when making a forecast, and for how many time steps in the future is it feasible to make forecasts?** For each model, we ran forecasts with different input windows and observed for both iterations that an input window taking into account the previous seven days resulted in the least amount of error on average for the best models. This is in line with the findings from related work [34]. Further, we found that doing this experiment for several configurations of the output window size did not change the seven-day trend.

  Forecasting one time-step yielded the least error, but further increases to the output window resulted in much higher error. However, we also observed that increasing the forecasting horizon from three and up showed little to no change in the RMSE score. Regarding the difficulties of forecasting only one time step and how much the error rises by only making the forecasting horizon slightly larger, we can conclude that forecasting more than one time step is not feasible and carries too much uncertainty with it.

- **Question 3: To what extent do multivariate forecasts produce better results than univariate forecasts?** In our experiments, on average across teams we observed a consistent positive effect on results by utilizing multivariate data. However, we observed exceptions. In the cases where the model either lacked the complexity to learn relationships between features, was not able to ignore unimportant features, or was not given enough data relative to its complexity, we saw that using multivariate data negatively affected performance compared to using univariate data. Still, these were exceptions, and for most data configurations and models, we saw a performance increase of up to 28% using a multivariate approach.

- **Question 4: What type of time series features are important when forecasting readiness to train?** Based on our data analysis in Chapter 4 and what we observed through our experiments, we determined that the most impactful features in forecasting readiness aside from the lagged readiness value were a mix of wellness and GPS-derived features. However, what features proved beneficial were reliant on the specific player's data. The features that consistently showed the most impact in our analysis included daily load, fatigue, Top Speed, HIR, and Total Distance.

- **Question 5: What is the viability of our selected use-cases and do they have potential to provide actionable data for coaches and players to use?** We demonstrated three use-cases using regression and classification and discussed their unique contribution in providing actionable data for players and coaches. Based on related work and our findings, we determined that easily interpretable statistics taking into account the unique data of each player are more favorable for a real-world use-case. Therefore, classification approaches classifying peaks or change in athletic development provides more useful insight than generating continuous RMSE values between one and ten. To further validate this statement, user studies need to be conducted.

We have answered our sub-question which allows us to go back and provide our solution to the main research question:

> *'What is a good approach to forecast readiness to train among professional female soccer players that result in actionable data for players and teams?'*

Our answer to the research question is an approach capturing important events or trends that consider each player's individual nature. Specifically, in terms of work presented in this thesis, it consists of combining several methods. Firstly, a time series approach predicting the next session that uses imputed data that can adequately contextualize missing data. Secondly, techniques ensuring optimal data configurations such as input window and feature selection tailored to the specific model and player. Thirdly, three-class classification such as classifying readiness peaks or positive, negative, or neutral changes in readiness as these use-cases reflect qualities that provide actionable data for training and game strategy decisions.

8.2 Other Contributions

Other contributions related to this thesis are as follows:

- **Source code:** All the source code created to implement our pipeline is accessible through a public repository on GitHub: https://github.com/simula/pmsys.

- **Dashboard:** The work in this thesis contributed to an interactive dashboard, deployed as a web application on Streamlit cloud, called "Soccer Dashboard": https://soccer-dashboard.simula.no

- **Paper submission (ongoing):** We are in the process of preparing a submission to a scientific venue (tentative title "Subjective readiness-to-play scores in soccer: how to predict and how to use"), in order to present the findings of this thesis in the form of a long paper.

8.3 Limitations of the Work

Despite previous conclusions, our findings suggest that player-based models might be a better approach. However, we could not follow up in-depth due to a lack of data. Another data-related limitation lies in the distribution. Because there are very few extreme data points, it is difficult for the models to reliably learn these patterns.

To understand what metrics have the greatest impact on an athlete's performance, conducting comprehensive surveys and studies using different metrics is needed. The perspective of this thesis is more centered around a computer science perspective rather than a sports science one and, therefore, might provide limited insight into the sports science aspect of this multidisciplinary field.

The field of machine learning and time series forecasting is enormous and constant development continuously produces better alternatives. The models used in this thesis only represent a fraction of relevant models used for time series prediction, which means that other better performing ML models may exist.

The use-cases we presented were based on previous work and observations during our data analysis. Other ways of predicting or representing readiness might yield more actionable data.

The reliability of the GPS data depends on the technology used to acquire them and the methods used to go from positional data to sports metrics like the average running speed of players. This process may cause an information loss where the sports metrics are not the exact original movements of the athletes. A similar issue is also present regarding the wellness reports provided by the players. If the players cannot answer reliably or the questions do not capture the athletic state of players, then the benefit of the wellness reports is diminished.

We mainly looked at how readiness moves over time for individual players. However, a more general overview time series perspective looking at the team's development over a longer time horizon might yield actionable data.

8.4 Future Work

The implementation of our pipeline has several possibilities to be used as a springboard for future work.

- **Surveys:** To better understand what will have a positive effect on soccer teams using ML, it can be beneficial to conduct surveys. These surveys can offer insight into what type of performance forecasting will provide actionable data.

- **Objective Data:** We have already observed the benefit of using objective data such as GPS-derived features in Chapter 4, 5 and 6. We also saw inconsistencies and a lack of contribution from the subjective wellness features. Therefore, focusing more on objective data might prove beneficial. Further, forecasting objective data might also lead to positive discoveries.
- **Online Learning and Deployment:** A reasonable next step is implementing a model in an online learning environment and deploying a passive ML analyzing tool on the PmSys app.
- **Different Use-Cases:** Rather than only focusing on wellness time series forecasting, it can prove beneficial to forecast strategies. For example, to see what movements or player position through GPS data leads to injuries or goals.
- **Other ML Models:** For our thesis we chose a selection of ML models which are described in Section 2.5. Choosing adequate models outside our selection can also lead to better results.
- **Model Fine-Tuning:** Fine-tune the models on a certain percentage of the data to the players that are being predicted.
- **Dynamic Parameter Selection:** Dynamically select the most relevant data configuration, meaning input window size and features, for each player and model type. This should increase results substantially based on our findings.

Overall, there are a plethora of possible avenues to investigate to further contribute to predicting athlete data to improve training conditions and derive game-changing strategies. These methods are not limited to only time series wellness forecasting in soccer but are also applicable to be integrated with other sports to derive other types of important statistics.

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 (Aug. 2019), pp. 1846–1855. DOI: 10.1111/sms.13535.

[2] Saad Albawi, Tareq Abed Mohammed and Saad Al-Zawi. 'Understanding of a convolutional neural network'. In: *2017 international conference on engineering and technology (ICET)*. Ieee. Akdeniz University, Antalya, Turkey, Aug. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[3] Arni Arnason, Stefan B Sigurdsson, Arni Gudmundsson, Ingar Holme, Lars Engebretsen and Roald Bahr. 'Physical fitness, injuries, and team performance in soccer'. In: *Medicine & Science in Sports & Exercise* 36.2 (2004), pp. 278–285. DOI: 10.1249/01.MSS.0000113478.92945.CA.

[4] Cornelius Arndt and Ulf Brefeld. 'Predicting the future performance of soccer players'. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 9.5 (Aug. 2016), pp. 373–382. DOI: 10.1002/sam.11321.

[5] James Bergstra, Daniel Yamins and David Cox. 'Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures'. In: *International conference on machine learning*. PMLR. Atlanta, USA, June 2013, pp. 115–123. URL: https://proceedings.mlr.press/v28/bergstra13.html.

[6] Chris M Bishop. 'Neural networks and their applications'. In: *Review of scientific instruments* 65.6 (June 1994), pp. 1803–1832. DOI: 10.1063/1.1144830.

[7] Bradley Camburn, Brock Dunlap, Tanmay Gurjar, Christopher Hamon, Matthew Green, Daniel Jensen, Richard Crawford, Kevin Otto and Kristin Wood. 'A systematic method for design prototyping'. In: *Journal of Mechanical Design* 137.8 (Aug. 2015), p. 081102. DOI: 10.1115/1.4030331.

[8] Bradley Camburn, Vimal Viswanathan, Julie Linsey, David Anderson, Daniel Jensen, Richard Crawford, Kevin Otto and Kristin Wood. 'Design prototyping methods: state of the art in strategies, tech-

niques, and guidelines'. In: *Design Science* 3 (Aug. 2017), e13. DOI: 10.1017/dsj.2017.10.

[9] Sean D Campbell and Francis X Diebold. 'Weather forecasting for weather derivatives'. In: *Journal of the American Statistical Association* 100.469 (Dec. 2005), pp. 6–16. DOI: 10.1198/016214504000001051.

[10] Mahil Carr and June Verner. 'Prototyping and software development approaches'. In: *Department of Information Systems, City University of Hong Kong, Hong Kong* (1997), pp. 319–338. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0b05add730e04843e234937a070f24b19efaadc3.

[11] 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*. San Francisco, USA, Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[12] 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 (July 2020), pp. 1454–1495. DOI: 10.1007/s10618-020-00701-z.

[13] Akshit J Dhruv, Reema Patel and Nishant Doshi. 'Python: the most advanced programming language for computer science applications'. In: *Proceedings of the international conference on culture heritage, education, sustainable tourism, and innovation technologies (CESIT 2020)*. 2021, pp. 292–299. URL: https://www.scitepress.org/Papers/2020/103079/103079.pdf.

[14] David A Dickey and Wayne A Fuller. 'Distribution of the estimators for autoregressive time series with a unit root'. In: *Journal of the American statistical association* 74.366a (1979), pp. 427–431. DOI: doi.org/10.1080/01621459.1979.10482531.

[15] Haipei Dong, Dakuo He and Fuli Wang. 'SMOTE-XGBoost using Tree Parzen Estimator optimization for copper flotation method classification'. In: *Powder Technology* 375 (July 2020), pp. 174–181. DOI: 10.1016/j.powtec.2020.07.065.

[16] Paul DuBois. *MySQL*. Pearson Education, 2008.

[17] Eyal Eliakim, Elia Morgulev, Ronnie Lidor and Yoav Meckel. 'Estimation of injury costs: financial damage of English Premier League teams' underachievement due to injuries'. In: *BMJ Open Sport & Exercise Medicine* 6.1 (May 2020), e000675. DOI: 10.1136/bmjsem-2019-000675.

[18] Kevin Fauvel, Tao Lin, Véronique Masson, Élisa Fromont and Alexandre Termier. 'XCM: An Explainable Convolutional Neural Network for Multivariate Time Series Classification'. In: *Mathematics* 9.23 (Dec. 2021), p. 3137. DOI: 10.3390/math9233137.

[19] Matt W Gardner and SR Dorling. 'Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences'. In: *Atmospheric environment* 32.14-15 (Aug. 1998), pp. 2627–2636. DOI: 10.1016/S1352-2310(97)00447-0.

[20] Isabelle Guyon, Jason Weston, Stephen Barnhill and Vladimir Vapnik. 'Gene selection for cancer classification using support vector machines'. In: *Machine learning* 46.1 (Jan. 2002), pp. 389–422. DOI: 10.1023/A:1012487302797.

[21] Gregory D Hager, Randal Bryant, Eric Horvitz, Maja Mataric and Vasant Honavar. 'Advances in artificial intelligence require progress across all of computer science'. In: *arXiv preprint arXiv:1707.04352* abs/1707.04352 (Feb. 2017). DOI: 10.48550/arXiv.1707.04352.

[22] Shona L Halson. 'Monitoring training load to understand fatigue in athletes'. In: *Sports medicine* 44.2 (Sept. 2014), pp. 139–147. DOI: 10.1007/s40279-014-0253-z.

[23] Pavel Hamet and Johanne Tremblay. 'Artificial intelligence in medicine'. In: *Metabolism* 69 (Mar. 2017), S36–S40. DOI: 10.1016/j.metabol.2017.01.011.

[24] Megan L Head, Luke Holman, Rob Lanfear, Andrew T Kahn and Michael D Jennions. 'The extent and consequences of p-hacking in science'. In: *PLoS biology* 13.3 (Mar. 2015), e1002106. DOI: 10.1371/journal.pbio.1002106.

[25] Julien Herzen, Francesco LÃ¤ssig, Samuele Giuliano Piazzetta, Thomas Neuer, LÃ©o Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan KoÅ›cisz, Dennis Bader, FrÃ©dÃ©rick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik and GaÃ«l Grosch. 'Darts: User-Friendly Modern Machine Learning for Time Series'. In: *Journal of Machine Learning Research* 23.124 (Jan. 2022), pp. 1–6. URL: http://jmlr.org/papers/v23/21-1177.html.

[26] Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin et al. 'Darts: User-friendly modern machine learning for time series'. In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 5442–5447. URL: https://dl.acm.org/doi/pdf/10.5555/3586589.3586713.

[27] Sepp Hochreiter and Jürgen Schmidhuber. 'Long short-term memory'. In: *Neural computation* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[28] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu and Douglas Eck. 'Music transformer'. In: *arXiv preprint arXiv:1809.04281* (Dec. 2018). DOI: 10.48550/arXiv.1809.04281.

[29] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller and François Petitjean. 'Inceptiontime: Finding alexnet for time series classification'. In: *Data Mining and Knowledge Discovery* 34.6 (Sept. 2020), pp. 1936–1962. DOI: 10.1007/s10618-020-00710-y.

[30] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal and Vitobha Munigala. 'Overview and importance of data quality for machine learning tasks'. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Long Beach , CA, USA, Aug. 2020, pp. 3561–3562. DOI: 10.1145/3394486.3406477.

[31] 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*. Utrecht, Netherlands, Dec. 2020, pp. 230–234. DOI: 10.1145/3395035.3425300.

[32] Håvard D Johansen, Svein Arne Pettersen, Pål Halvorsen and Dag Johansen. 'Combining Video and Player Telemetry for Evidence-based Decisions in Soccer.' In: *icSPORTS*. Vilamoura, Algarve, Portugal, 2013, pp. 197–205. URL: https://home.simula.no/~paalh/publications/files/icSport2013.pdf.

[33] Donald T Kirkendall and Jiri Dvorak. 'Effective injury prevention in soccer'. In: *The physician and sportsmedicine* 38.1 (Mar. 2015), pp. 147–157. DOI: 10.3810/psm.2010.04.1772.

[34] Siarhei Kulakou, Nourhan Ragab, Cise Midoglu, Matthias Boeker, Dag Johansen, Michael A Riegler and Pål Halvorsen. 'Exploration of Different Time Series Models for Soccer Athlete Performance Prediction'. In: *Engineering Proceedings* 18.1 (June 2022), p. 37. DOI: 10.3390/engproc2022018037.

[35] Bryan Lim, Sercan Ö Arık, Nicolas Loeff and Tomas Pfister. 'Temporal fusion transformers for interpretable multi-horizon time series forecasting'. In: *International Journal of Forecasting* 37.4 (2021), pp. 1748–1764. DOI: 10.1016/j.ijforecast.2021.03.012.

[36] Joao Lourencco, Elvio Rubio Gouveia, Hugo Sarmento, Andreas Ihle, Tiago Ribeiro, Ricardo Henriques, Francisco Martins, Cintia Francca, Ricardo Maia Ferreira, Luis Fernandes et al. 'Relationship between Objective and Subjective Fatigue Monitoring Tests in Professional Soccer'. In: *International Journal of Environmental Research and Public Health* 20.2 (Jan. 2023), p. 1539. DOI: 10.3390/ijerph20021539.

[37] Scott M Lundberg and Su-In Lee. 'A unified approach to interpreting model predictions'. In: *Advances in neural information processing systems* 30 (Dec. 2017), pp. 4765–4774. URL: https://proceedings.neurips.

cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[38] Wes McKinney. 'Data Structures for Statistical Computing in Python'. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. Austin, Texas, 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[39] 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.

[40] Thu Nguyen, Khoi Minh Nguyen-Duy, Duy Ho Minh Nguyen, Binh T Nguyen and Bruce Alan Wade. 'Dper: Direct parameter estimation for randomly missing data'. In: *Knowledge-Based Systems* 240 (Jan. 2022), p. 108082. DOI: 10.1016/j.knosys.2021.108082.

[41] Ignacio Oguiza. *tsai - A state-of-the-art deep learning library for time series and sequential data*. Github. 2022. URL: https://github.com/timeseriesAI/tsai.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga et al. 'Pytorch: An imperative style, high-performance deep learning library'. In: *Advances in neural information processing systems* 32 (Dec. 2019), pp. 8026–8037. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (Dec. 2011), pp. 2825–2830. URL: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https:/.

[44] 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 (July 2018), p. 866. DOI: 10.3389/fphys.2018.00866.

[45] W Nicholson Price and I Glenn Cohen. 'Privacy in the age of medical big data'. In: *Nature medicine* 25.1 (Jan. 2019), pp. 37–43. DOI: 10.1038/s41591-018-0272-7.

[46] Nourhan Ragab. 'Soccer athlete performance prediction using time series analysis'. MA thesis. OsloMet-storbyuniversitetet, 2022. URL: http://home.simula.no/~paalh/students/NourhanRagab-OsloMet-2022.pdf.

[47] Mauricio Reyes, Raphael Meier, Sérgio Pereira, Carlos A Silva, Fried-Michael Dahlweid, Hendrik von Tengg-Kobligk, Ronald M Summers and Roland Wiest. 'On the interpretability of artificial intelligence in radiology: challenges and opportunities'. In: *Radiology: artificial intelligence* 2.3 (May 2020), e190043. DOI: 10.1148/ryai.2020190043.

[48] Frank Rosenblatt. 'The perceptron: a probabilistic model for information storage and organization in the brain.' In: *Psychological review* 65.6 (Nov. 1958), p. 386. DOI: 10.1037/h0042519.

[49] 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 (July 2018), e0201264. DOI: 10.1371/journal.pone.0201264.

[50] Alex Rubinsteyn and Sergey Feldman. *fancyimpute: An Imputation Library for Python*. Version 0.7.0. 2016. URL: https://github.com/iskandr/fancyimpute.

[51] Jeffrey S Saltz and Neil Dewar. 'Data science ethical considerations: a systematic literature review and proposed project framework'. In: *Ethics and Information Technology* 21 (Mar. 2019), pp. 197–208. DOI: 10.1007/s10676-019-09502-5.

[52] Arthur L Samuel. 'Some studies in machine learning using the game of checkers. II—Recent progress'. In: *IBM Journal of research and development* 11.6 (Nov. 1967), pp. 601–617. DOI: 10.1147/rd.116.0601.

[53] 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 (Jan. 2015), p. 137. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4306765/pdf/jssm-14-137.pdf.

[54] Skipper Seabold and Josef Perktold. 'statsmodels: Econometric and statistical modeling with python'. In: *9th Python in Science Conference*. Austin, Texas, July 2010, pp. 92–96. URL: https://pdfs.semanticscholar.org/3a27/6417e5350e29cb6bf04ea5a4785601d5a215.pdf.

[55] Xiaogang Su, Xin Yan and Chih-Ling Tsai. 'Linear regression'. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.3 (Aug. 2012), pp. 275–294. DOI: 10.1002/wics.1198.

[56] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[57] Poojan Thakkar and Manan Shah. 'An assessment of football through the lens of data science'. In: *Annals of Data Science* 8 (Mar. 2021), pp. 823–836. DOI: 10.1007/s40745-021-00323-2.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017), pp. 5998–6008. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[59] H Wang, ZeZXeZBePJ Lei, X Zhang, B Zhou and J Peng. 'Machine learning basics'. In: *Deep learning* (2016), pp. 98–164. URL: http://whdeng.cn/Teaching/PPT_01_Machine%5C%20learning%5C%20Basics.pd.

[60] 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. Dublin, Ireland, Sept. 2019, pp. 1–6. DOI: 10.1109/CBMI.2019.8877406.

[61] Chris Wing. 'Monitoring athlete load: Data collection methods and practical recommendations'. In: *Strength & Conditioning Journal* 40.4 (Aug. 2018), pp. 26–39. DOI: 10.1519/SSC.0000000000000384.

[62] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer and Peter Vajda. 'Visual transformers: Token-based image representation and processing for computer vision'. In: *arXiv preprint arXiv:2006.03677* abs/2006.03677 (2020). DOI: 10.48550/arXiv.2006.03677.

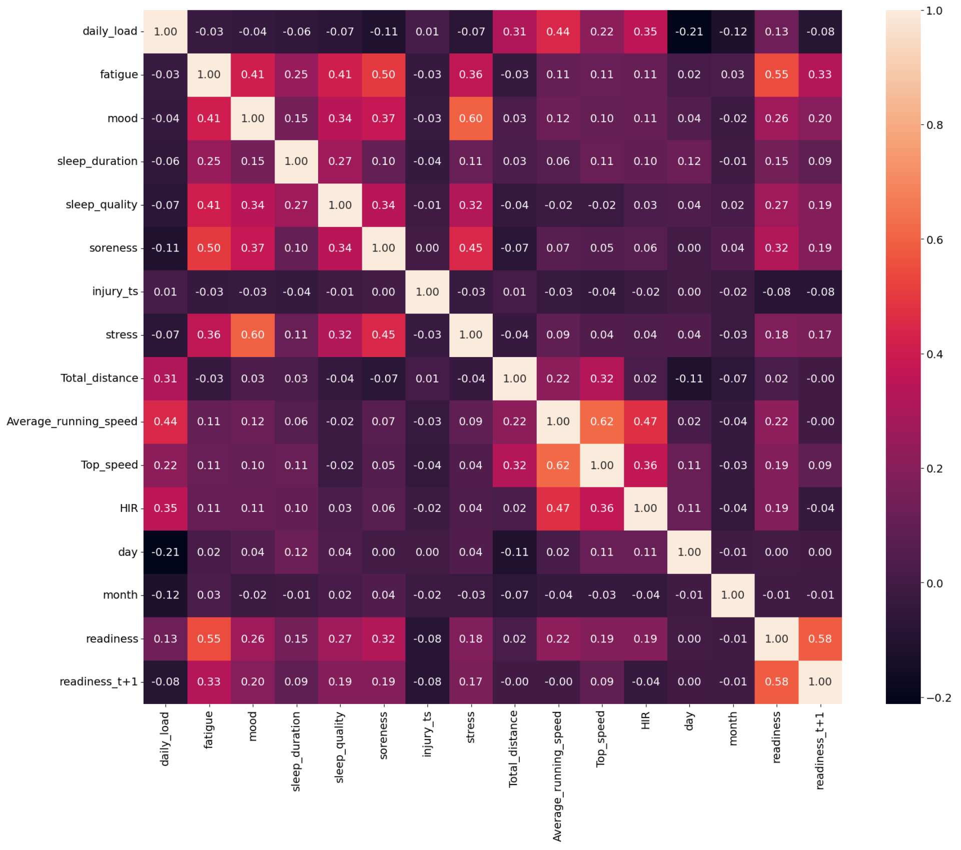
[63] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty and Carsten Eickhoff. 'A transformer-based framework for multivariate time series representation learning'. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. Singapore, Aug. 2021, pp. 2114–2124. DOI: 10.1145/3447548.3467401.

Chapter 9

Appendix

9.1 Correlation Matrices After Imputation

Figures 9.1 and 9.2 is used in Section 4.5 and show the correlation matrix between all features after imputation for teams A and B.



	daily_load	fatigue	mood	sleep_duration	sleep_quality	soreness	injury_ts	stress	Total_distance	Average_running_speed	Top_speed	HIR	day	month	readiness	readiness_t+1
daily_load	1.00	-0.03	-0.04	-0.06	-0.07	-0.11	0.01	-0.07	0.31	0.44	0.22	0.35	-0.21	-0.12	0.13	-0.08
fatigue	-0.03	1.00	0.41	0.25	0.41	0.50	-0.03	0.36	-0.03	0.11	0.11	0.11	0.02	0.03	0.55	0.33
mood	-0.04	0.41	1.00	0.15	0.34	0.37	-0.03	0.60	0.03	0.12	0.10	0.11	0.04	-0.02	0.26	0.20
sleep_duration	-0.06	0.25	0.15	1.00	0.27	0.10	-0.04	0.11	0.03	0.06	0.11	0.10	0.12	-0.01	0.15	0.09
sleep_quality	-0.07	0.41	0.34	0.27	1.00	0.34	-0.01	0.32	-0.04	-0.02	-0.02	0.03	0.04	0.02	0.27	0.19
soreness	-0.11	0.50	0.37	0.10	0.34	1.00	0.00	0.45	-0.07	0.07	0.05	0.06	0.00	0.04	0.32	0.19
injury_ts	0.01	-0.03	-0.03	-0.04	-0.01	0.00	1.00	-0.03	0.01	-0.03	-0.04	-0.02	0.00	-0.02	-0.08	-0.08
stress	-0.07	0.36	0.60	0.11	0.32	0.45	-0.03	1.00	-0.04	0.09	0.04	0.04	0.04	-0.03	0.18	0.17
Total_distance	0.31	-0.03	0.03	0.03	-0.04	-0.07	0.01	-0.04	1.00	0.22	0.32	0.02	-0.11	-0.07	0.02	-0.00
Average_running_speed	0.44	0.11	0.12	0.06	-0.02	0.07	-0.03	0.09	0.22	1.00	0.62	0.47	0.02	-0.04	0.22	-0.00
Top_speed	0.22	0.11	0.10	0.11	-0.02	0.05	-0.04	0.04	0.32	0.62	1.00	0.36	0.11	-0.03	0.19	0.09
HIR	0.35	0.11	0.11	0.10	0.03	0.06	-0.02	0.04	0.02	0.47	0.36	1.00	0.11	-0.04	0.19	-0.04
day	-0.21	0.02	0.04	0.12	0.04	0.00	0.00	0.04	-0.11	0.02	0.11	0.11	1.00	-0.01	0.00	0.00
month	-0.12	0.03	-0.02	-0.01	0.02	0.04	-0.02	-0.03	-0.07	-0.04	-0.03	-0.04	-0.01	1.00	-0.01	-0.01
readiness	0.13	0.55	0.26	0.15	0.27	0.32	-0.08	0.18	0.02	0.22	0.19	0.19	0.00	-0.01	1.00	0.58
readiness_t+1	-0.08	0.33	0.20	0.09	0.19	0.19	-0.08	0.17	-0.00	-0.00	0.09	-0.04	0.00	-0.01	0.58	1.00

Figure 9.1: Pairwise correlation matrix of our data after imputation for Team A.

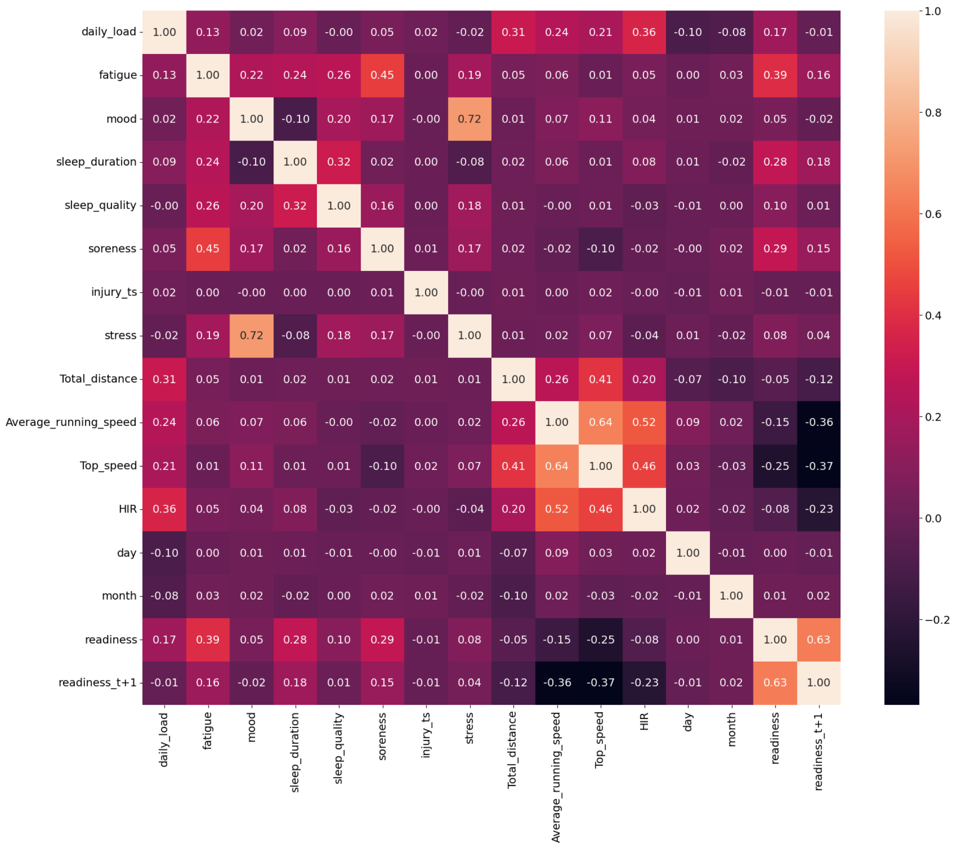
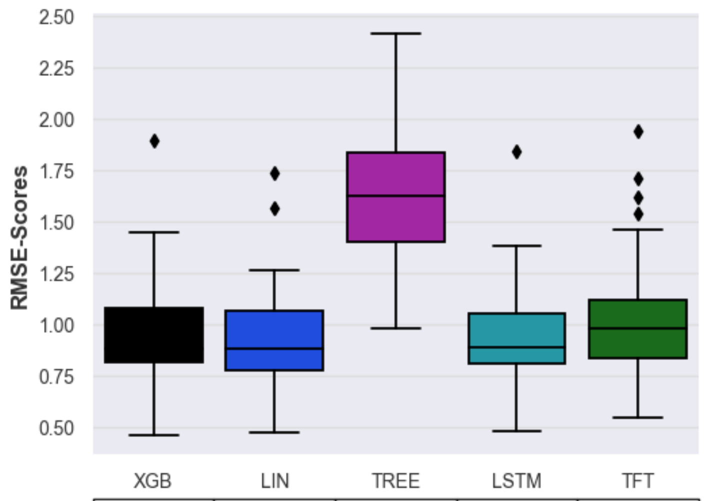


Figure 9.2: Pairwise correlation matrix of our data after imputation for Team B.

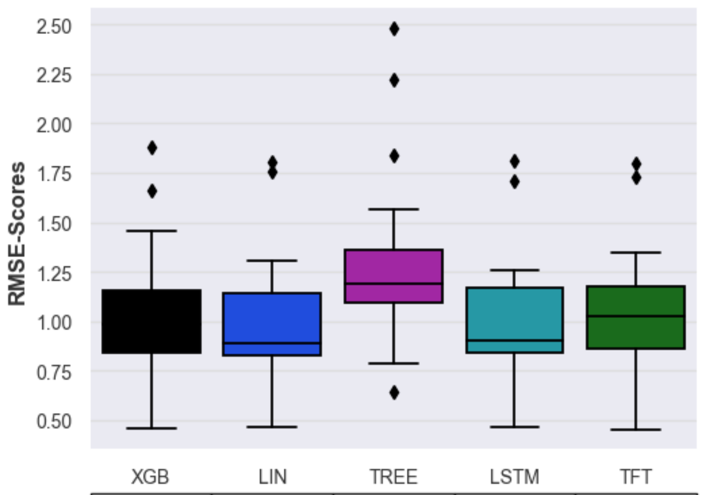
9.2 Boxplots Training on Both Teams Predicting Players From Either Team A or B

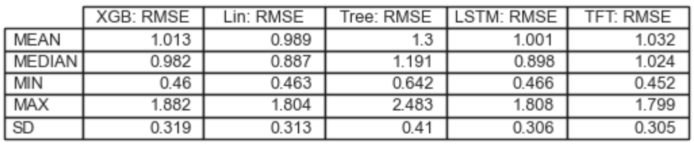
Figures 9.3, 9.4, 9.5, and 9.6 are used in Section 5.1.10 showing how training on both teams impact readiness RMSE scores for both univariate and multivariate data.



	XGB: RMSE	Lin: RMSE	Tree: RMSE	LSTM: RMSE	TFT: RMSE
MEAN	0.964	0.946	1.635	0.964	1.036
MEDIAN	0.882	0.883	1.624	0.891	0.978
MIN	0.464	0.474	0.98	0.48	0.545
MAX	1.899	1.74	2.417	1.843	1.943
SD	0.316	0.286	0.338	0.293	0.36

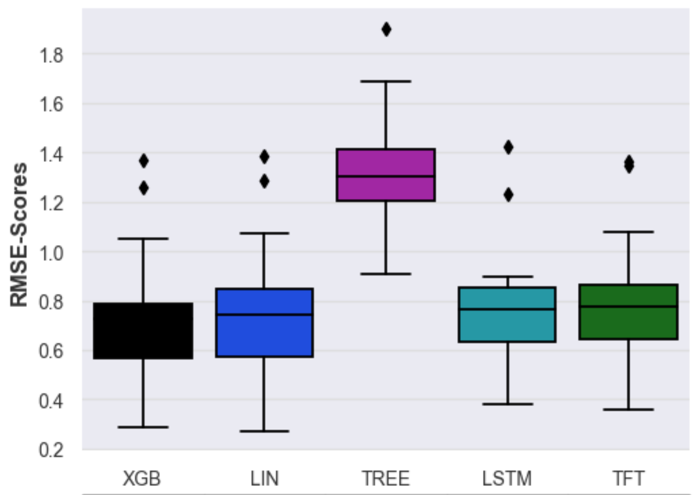
Figure 9.3: RMSE training on both team A and B predicting players from A (multivariate).





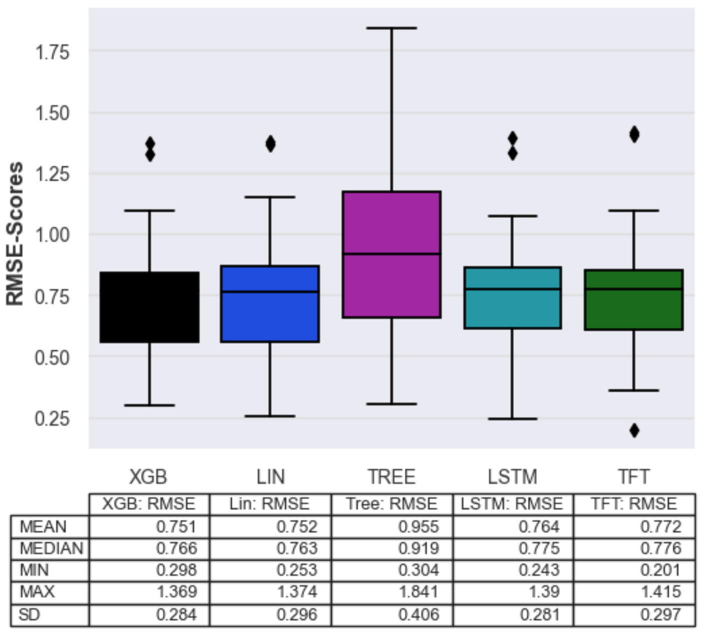
	XGB: RMSE	Lin: RMSE	Tree: RMSE	LSTM: RMSE	TFT: RMSE
MEAN	1.013	0.989	1.3	1.001	1.032
MEDIAN	0.982	0.887	1.191	0.898	1.024
MIN	0.46	0.463	0.642	0.466	0.452
MAX	1.882	1.804	2.483	1.808	1.799
SD	0.319	0.313	0.41	0.306	0.305

Figure 9.4: RMSE training on both team A and B predicting players from A (univariate).



	XGB: RMSE	Lin: RMSE	Tree: RMSE	LSTM: RMSE	TFT: RMSE
MEAN	0.728	0.742	1.319	0.764	0.787
MEDIAN	0.727	0.74	1.301	0.762	0.776
MIN	0.287	0.273	0.91	0.382	0.358
MAX	1.368	1.383	1.901	1.421	1.364
SD	0.27	0.273	0.228	0.237	0.275

Figure 9.5: RMSE training on both team A and B predicting players from B (multivariate).

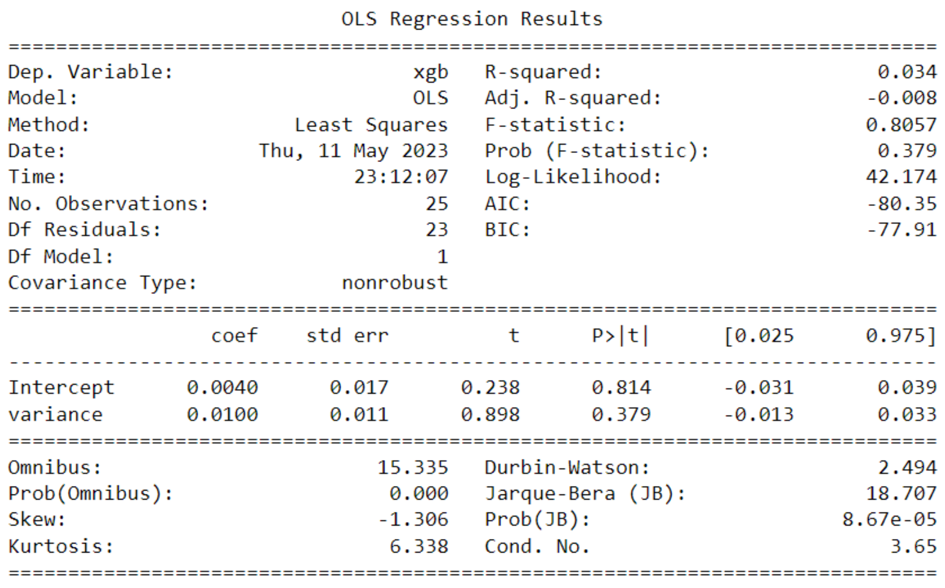


	XGB: RMSE	Lin: RMSE	Tree: RMSE	LSTM: RMSE	TFT: RMSE
MEAN	0.751	0.752	0.955	0.764	0.772
MEDIAN	0.766	0.763	0.919	0.775	0.776
MIN	0.298	0.253	0.304	0.243	0.201
MAX	1.369	1.374	1.841	1.39	1.415
SD	0.284	0.296	0.406	0.281	0.297

Figure 9.6: RMSE training on both team A and B predicting players from B (univariate).

9.3 OLS-analysis

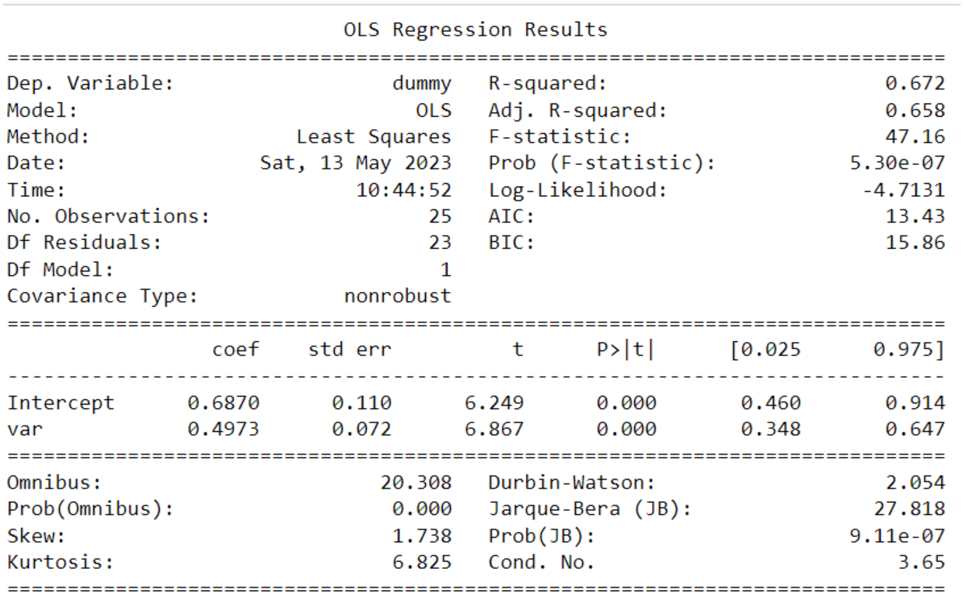
Figure 9.7 is used in Section 5.1.8, and is a regression analysis showing the impact variance has on RMSE difference between using multivariate and univariate data.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    xgb   R-squared:                       0.034
Model:                            OLS   Adj. R-squared:                 -0.008
Method:                 Least Squares   F-statistic:                    0.8057
Date:                Thu, 11 May 2023   Prob (F-statistic):              0.379
Time:                        23:12:07   Log-Likelihood:                 42.174
No. Observations:                  25   AIC:                            -80.35
Df Residuals:                      23   BIC:                            -77.91
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0040      0.017      0.238      0.814      -0.031       0.039
variance       0.0100      0.011      0.898      0.379      -0.013       0.033
==============================================================================
Omnibus:                       15.335   Durbin-Watson:                   2.494
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               18.707
Skew:                          -1.306   Prob(JB):                     8.67e-05
Kurtosis:                       6.338   Cond. No.                         3.65
==============================================================================
```

Figure 9.7: OLS analysis showing how variance impact model error.

Figure 9.8 is used in Section 5.1.6, and is a regression analysis showing the impact variance has on RMSE in respect to dummy model performance.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  dummy   R-squared:                       0.672
Model:                            OLS   Adj. R-squared:                  0.658
Method:                 Least Squares   F-statistic:                     47.16
Date:                Sat, 13 May 2023   Prob (F-statistic):           5.30e-07
Time:                        10:44:52   Log-Likelihood:                -4.7131
No. Observations:                  25   AIC:                             13.43
Df Residuals:                      23   BIC:                             15.86
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.6870      0.110      6.249      0.000       0.460       0.914
var            0.4973      0.072      6.867      0.000       0.348       0.647
==============================================================================
Omnibus:                       20.308   Durbin-Watson:                   2.054
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               27.818
Skew:                           1.738   Prob(JB):                     9.11e-07
Kurtosis:                       6.825   Cond. No.                         3.65
==============================================================================
```

Figure 9.8: OLS analysis showing how variance increases RMSE using a dummy model predicting the mean.