Master's Thesis

AI-Driven Actigraphy Data Reconstruction

by Rodrigo Santos da Silva

advised by
Prof. Ph.D. Thiago Damasceno Cordeiro
Prof. Ph.D. Tiago Gomes de Andrade

Federal University of Alagoas
Computing Institute
Maceió, Alagoas
December 12, 2024

FEDERAL UNIVERSITY OF ALAGOAS
Computing Institute

AI-DRIVEN ACTIGRAPHY DATA RECONSTRUCTION

Master's thesis submitted to the Computing Institute from Federal University of Alagoas as a partial requirement to obtain the master's degree in Informatics.

Rodrigo Santos da Silva

Advisor: Prof. Ph.D. Thiago Damasceno Cordeiro
Advisor: Prof. Ph.D. Tiago Gomes de Andrade

Maceió, Alagoas
December 12, 2024

Folha de Aprovação

RODRIGO SANTOS DA SILVA

RECONSTRUÇÃO DE DADOS DE ACTIGRAFIA ORIENTADOS POR IA

AI-DRIVEN ACTIGRAPHY DATA RECONSTRUCTION

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 12 de dezembro de 2024.

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Helena Cristina Pimentel do Vale – CRB4 - 661

S586a Silva, Rodrigo Santos da.
AI-driven actigraphy data reconstruction / Rodrigo Santos da Silva. – 2025.
68 f.: il.

Orientador: Thiago Damasceno Cordeiro, Tiago Gomes de Andrade.
Dissertação (mestrado em Informática) – Universidade Federal de Alagoas, Instituto de Computação. Maceió, 2024.

Texto em Inglês.
Bibliografia: f. 58-63.

1. Actigrafia. 2. Aprendizado de maquina. 3. Aprendizado profundo. 4. M10. 5. PIM. I. Título.

CDU: 004.8

Acknowledgments

I would like to express my gratitude to Prof Thiago Cordeiro for all the support, not only in this thesis but in all my graduation time, for allowing me to become a better professional, a better researcher and showing me the kind of professor that I want to be. A Special thanks to Prof. Tiago Andrade and Prof. Lívia Goes Gitaí, who helped me with guidance and knowledge. Additionally, I would like to express my gratitude to Prof. Rafael Mello and Prof. Bruno Pimentel for generously dedicating their time and participating in the chair.

I want to thank my family, in special my parents Ivone and Roberval, my stepmother Marlene, and my siblings, Vitória and Lorenzo, for always being there for me and supporting my studies. Thank you for showing me the importance of education.

There are no words to describe how Fernanda was important in this whole process, as a wonderful partner, friend, and coworker. Thank you for always being there for me. Meeting you in college was the greatest gift.

Also, a special thanks to all my friends from IFAL that have been by my side, even though we are not studying together anymore. A special thanks to Aguida, Clara, Gabriel, Haniel, Juliane, Karen, Luísa, Henrique, Márcio Henrique, Michael, João Falcão, and Pedro Henrique.

Lastly, a special thanks to my friends Bruno, Vívian, Sophia, Arthur, José Neto, Maria Eduarda, Lucas, Larissa, Eduarda Chagas, Gabriel Germano, Rafael, Laryssa, Álvaro, Vanessa, and Kevin for the pleasure of working, having fun, and learning with you all.

And thank you, dear reader, for having an interest in this work.

One learns from books and example only that certain things can be done. Actual learning requires that you do those things.
Frank Herbert

Resumo

A actigrafia é uma técnica não invasiva que utiliza um dispositivo de pulso para monitorar padrões de sono e ritmos circadianos, fornecendo dados valiosos para o diagnóstico de distúrbios do sono, como insônia e hipersonia. No entanto, lacunas nos dados de actigrafia frequentemente ocorrem quando o dispositivo está “fora do pulso”. Este estudo explora o uso de técnicas avançadas de Machine Learning e Deep Learning para imputar dados ausentes, visando melhorar a precisão e a confiabilidade das análises. O principal objetivo é desenvolver pipelines robustos de imputação de dados, utilizando uma variedade de algoritmos, incluindo modelos de regressão (Random Forests, XGBoost e Support Vector Regression), com estratégias de otimização de hiperparâmetros, como Grid Search e Randomized Search, e autoencoders otimizados pelo Keras Tuner. As principais estratégias de filtragem incluem o Dynamic Time Warping, aproveitando a variável M10, que representa as 10 horas de maior atividade diária, além de considerar os dados em torno dos períodos ”fora do pulso”. A metodologia pré-processa conjuntos de dados de actigrafia do dispositivo ActTrust (Condor), incorporando variáveis como temperatura, exposição à luz e métricas de atividade do paciente. Resultados preliminares demonstram que os modelos de autoencoder superam os métodos tradicionais na imputação de dados ausentes, reduzindo significativamente o erro médio quadrático. Modelos de ensemble, ajustados para padrões específicos de atividade, melhoram ainda mais a precisão preditiva, conforme validado por testes estatísticos, como o teste de Wilcoxon. Esses achados destacam o potencial desses modelos para aprimorar a prática clínica, permitindo avaliações personalizadas do sono e apoiando intervenções direcionadas.

***Palavras-chave*: *Actigrafia*; *Aprendizado de Máquina*; *Aprendizado Profundo*; *M10*; *PIM*.**

Abstract

Actigraphy is a non-invasive technique that uses a wrist-worn device to track sleep patterns and circadian rhythms, providing valuable data for diagnosing sleep disorders such as insomnia and hypersomnia. However, gaps in actigraphy data often occur when the device is ”off-wrist.” This study explores the use of advanced Machine Learning and Deep Learning techniques to impute missing data, aiming to improve the accuracy and reliability of the analyses. The main objective is to develop robust data imputation pipelines utilizing a range of algorithms, including regression models (Random Forests, XGBoost, and Support Vector Regression), with hyperparameter optimization strategies such as Grid Search and Randomized Search and autoencoders optimized by Keras Tuner. Along with these regression strategies, we have vital data filtering strategies, including Dynamic Time Warping, leveraging the M10 variable, which represents the 10 hours of highest daily activity, and considering data surrounding off-wrist periods. In this study, the methodology preprocesses actigraphy datasets from the ActTrust device, incorporating variables such as temperature, light exposure, and activity metrics. Preliminary results demonstrate that autoencoder models outperform traditional methods in imputing missing data, significantly reducing Mean Squared Error. Combined models tailored to specific activity patterns further enhance predictive accuracy, as validated by statistical tests such as the Wilcoxon test. These findings highlight the potential of these models to improve clinical practice by enabling personalized sleep assessments and supporting targeted interventions.

***Keywords*: *Actigraphy*; *Machine Learning*; *Deep Learning*; *M10*; *PIM*.**

List of Figures

List of Tables

List of Acronyms

AI	- *Artificial Intelligence.*
DL	- *Deep Learning.*
DTW	- *Dynamic Time Warping.*
IS	- *Interdaily Stability.*
IV	- *Intradaily Variability.*
KNN	- *K-Nearest Neighbor.*
L5	- *Least 5.*
M10	- *Most 10.*
MAE	- *Mean Absolute Error.*
MASE	- *Mean Absolute Scaled Error.*
ML	- *Machine Learning.*
MSE	- *Mean Squared Error.*
NLP	- *Natural Language Processing.*
ON	- *On-Wrist.*
OW	- *Off-Wrist.*
PIM	- *Proportional Integral Mode.*
PIM$_n$	- *Normalized Proportional Integral Mode.*
PSG	- *Polysomnography.*
RA	- *Relative Amplitude.*
R^2	- *Coefficient of Determination.*

RFC - *Random Forest Classifier*.

RMSE - *Root Mean Squared Error*.

TAT - *Time Above Threshold*.

TAT$_n$ - *Normalized Time Above Threshold*.

ZCM - *Zero Crossing Mode*.

ZCM$_n$ - *Normalized Zero Crossing Mode*.

Contents

Chapter 1

Introduction

Sleep constitutes a fundamental pillar of our holistic well-being, exerting significant influence across multiple dimensions of our lives beyond mere repose. A pivotal function of sleep is the facilitation of restoration and recuperation, affording our physiological framework the opportunity for tissue, muscular, organic repair, and revitalization (Anafi et al., 2013). Furthermore, sleep assumes a critical role in cognitive processes, augmenting memory consolidation, problem-solving acumen, creative faculties, and aptitude for decision-making. It also profoundly impacts emotional equilibrium, regulating mood, mitigating irritability, and fostering emotional equilibrium (Scott et al., 2021). Additionally, sleep is indispensable for physical vitality, contributing to energy preservation, modulation of hormones governing appetite and metabolic functions, and bolstering immune defenses against disorders and infections (Rogers et al., 2024).

Moreover, optimal sleep quality manages stress levels, bolstering resilience and overall psychological well-being. Finally, adequate sleep enhances physical performance, refining coordination, reaction time, and athletic capabilities. In sum, prioritizing adequate and rejuvenating sleep is imperative for optimizing our physical, mental, and emotional welfare (Charest and Grandner, 2020).

Considering all the elements presented in the previous paragraphs, many methods were developed to monitor sleep and analyze the quality of sleep. One widely used method for tracking sleep and activity in real-world settings is actigraphy, a non-invasive technique to identify individual sleep and wake cycles over a predetermined timeframe. It relies on using a portable apparatus known as an actigraph, typically worn on the non-dominant wrist of adults (and occasionally positioned on the hip or ankle for children), resembling a wristwatch. This apparatus aids in the detection of circadian rhythm disruptions, encompassing conditions such as hypersomnia, insomnia, circadian rhythm disorders, excessive somnolence, as well as advanced or delayed sleep phase syndromes (Grandner and Rosenberger, 2019).

Its fundamental constituents comprise an accelerometer for recording movement, a data storage unit, and a programmable timing mechanism. Specific models may incorporate additional components, such as spectrophotometers for discerning various luminosity levels and thermometers for monitoring patient and environmental temperatures (Zambotti et al., 2019). These supplementary components provide specialists with valuable insights into the patient's condition at specific junctures, thereby aiding in identifying sleep disorders. Leveraging the data from the device, specialists can ascertain the onset and cessation of sleep, the latency period preceding sleep onset, total sleep duration, and the duration of wakefulness after sleep onset during nocturnal periods.

Actigraphy may be employed autonomously or in tandem with supplementary sleep evaluation methods, such as sleep logs or overnight sleep studies (Garefelt et al., 2022). While it does not attain the benchmark status attributed to polysomnography (PSG), actigraphy presents a viable alternative in circumstances where PSG proves impractical, notably with pediatric patients or individuals necessitating tailored sleep assessments. This viability is owing to its capacity to emulate a familiar wearing pattern akin to a watch, enhancing patient compliance and comfort.

Nonetheless, patients need to wear the device for prolonged durations poses several challenges, encompassing potential device damage, improper utilization, and instances of device removal from the wrist, commonly denoted as "off-wrist" (OW) occurrences. While specialists advise patients against removing the device until the data collection process concludes, specific individuals may opt to do so under varying circumstances, such as engaging in sports activities (due to concerns regarding potential damage to the device), during showering, and, perhaps most significantly, during specific periods of sleep.

In days where OW intervals exceed 4 hours, excluding that day from the analysis and focusing on the remaining days with less than 4 hours of OW periods is recommended (Tonon et al., 2022). Attempting to reconstruct days with more than 4 hours of off-wrist time is comparable to trying to infer data patterns with a significant amount of missing information, compromising the imputation's accuracy. This extended off-wrist duration introduces substantial uncertainty, as prolonged periods without data reduce the reliability of pattern recognition algorithms and make it difficult to maintain the temporal continuity necessary for accurate imputations. Moreover, including such days in the analysis can introduce noise into the model, leading to less robust predictions and potentially skewing the overall results. By excluding days with excessive off-wrist time, we prioritize the integrity of the dataset, ensuring that the imputation process can rely on a sufficient amount of valid, continuous data to generate more accurate reconstructions.

Moreover, attention must be directed towards the identification process for OW intervals, as not all devices incorporate a structured mechanism to discern such instances. Even when present, these mechanisms may not provide explicit indication to specialists regarding the nature of a specific interval (i.e., whether it constitutes an OW moment or not). For instance, with the ActTrust (Condor actigraph) and ActStudio (its accompanying software), it is customary to interpolate OW intervals with 0 values (Fekedulegn et al., 2020). However, it's imperative to note that values of 0 hold significance within the data analysis spectrum for certain variables, as exemplified by the Proportional Integral Mode (PIM), which signifies the level of activity exhibited by the patient during a given moment.

In this context, developing a tool capable of performing data imputation for patients exhibiting OW moments within their actigraphy examination results could aid specialists in diagnosing sleep disorders or other conditions associated with sleep quality. Such a tool must be able to adapt to each patient's unique realities and patterns, as individuals inherently exhibit distinct activity patterns attributable to variations in their daily routines and lifestyles. By accommodating these individualized patterns, the tool can enhance its efficacy in accurately assessing sleep parameters and facilitating tailored interventions to address specific patient needs.

1.1 Objectives

The primary objective of this study is to present and test different approaches to perform data imputation in actigraphy data using artificial intelligence techniques (mainly regression techniques, with Machine Learning and Deep Learning approaches), with the central goal of accurately imputing missing data within applicable scenarios while adapting to the unique data patterns of individual patients.

1.2 Specific Objectives

For the primary goal, the following specific objectives will be contemplated.

1. Examine the efficacy of employing machine learning regression models and autoencoders in the data imputation process.
2. Perform a comprehensive analysis of the current variables within the datasets.
3. Conduct pre-processing and feature transformation procedures on the existing data.
4. Construct AI models using Machine Learning and Deep Learning approaches with hyperparameter tunning techniques to optimize predictive performance across tested models for individual patients.

1.3 Applications

The imputation of actigraphy data holds the potential to aid specialists in diagnosing or predicting certain illnesses related to the circadian cycle. Additionally, by utilizing distinct groups with predefined diagnoses of specific diseases, ML and DL models can be tailored for imputing data in targeted scenarios, where each model demonstrates superior performance for the corresponding group.

In practical applications, such as web-based platforms integrated with distributed systems, deploying these models within an adaptable AI pipeline facilitates seamless replication and scalability across diverse healthcare contexts. This deployment enables healthcare providers to efficiently utilize imputed actigraphy data for informed decision-making, personalized treatment planning, and monitoring of patient progress over time. By harnessing the potential of targeted modeling in data imputation, healthcare professionals can unlock valuable insights from actigraphy data, improving patient care and managing circadian-related illnesses.

1.4 Structure

The introductory chapter of the thesis will delineate the subject matter under investigation. Subsequently, the second chapter will provide an in-depth exploration of the background informing the methodologies elucidated within this thesis. This exploration elucidates actigraphy data, a spectrum of ML and DL algorithms, and the procedural framework to perform the imputation and some state of rate articles. In the third chapter, a comprehensive exposition of the methodology will ensue, outlining the procedural steps to derive the outcomes. The fourth chapter will present the findings obtained from the study conducted. Lastly, the conclusion chapter presents critical insights and considerations from the research endeavor.

Chapter 2

Theoretical Foundation

This section aims to provide readers with a solid foundation for understanding each step of data imputation and concepts related to actigraphy. Key topics include Machine Learning models, particularly regressors, and statistical metrics for evaluating model performance on our datasets. We will also discuss techniques for handling and comparing data vectors, such as Dynamic Time Warping, and methods for transforming and normalizing data to compatible ranges. We will cover strategies for selecting the optimal model within a specified range of hyperparameters. Additionally, we will explore autoencoders, a technique used in the imputation process. Regarding actigraphy, we will discuss the variables in our dataset, their importance in our context, and the standard actigraphy metrics used to evaluate the circadian cycle.

2.1 Actigraphy

The first actigraphy devices were developed by J.S. Szymanski in the early 1970s and have been continuously improved over the years to reach the current state of the art. As mentioned, modern actigraphy devices, such as accelerometers, include movement detectors and have enough memory to record data for several weeks. These devices sample movement multiple times per second and store the data for later analysis. Researchers use computer programs, like ActStudio, which comes with the ActTrust device we used in this study, to analyze the data. As showed by Ancoli-Israel et al. (2003), these programs calculate levels of activity/inactivity, rhythm parameters (such as amplitude or acrophase), and sleep/wake parameters (including total sleep time, percentage of time spent asleep, total wake time, percentage of time spent awake, and the number of awakenings).

Different actigraphy devices collect varying data types, including documentation, luminosity, temperature, and others (Jumabhoy et al., 2019). These variables directly impact how researchers evaluate sleep quality and assess the patient's circadian cycle definition. Researchers commonly use actigraphy along with other methods such as PSG (Kaplan et al., 2012), Sleep Diaries (Mazza et al., 2020), and Questionnaires (Brink-Kjaer et al., 2023) since actigraphy by itself does not have the same amount of information such as PSG, which is the gold pattern for this area. But, the fact that it is usually not used to perform diagnoses doesn't subtract the impact that actigraphy can have in the process of facilitating collecting patient data since PSG requires the participant to stay overnight in a sleep lab with various sensors attached to their body, actigraphy allows the individual to maintain their routine while wearing a small, wristwatch-like device, which is especially useful for studying sleep patterns in real-world settings.

Furthermore, Actigraphy is much more affordable than PSG, making it accessible for large-scale population studies or long-term monitoring in clinical practice. It's often used for initial screenings for sleep disorders or to track the effectiveness of treatments over time (Lehrer et al., 2022). Actigraphy is especially useful for populations where sleep patterns are highly variable or complex to monitor in a lab setting, such as children (Schoch et al., 2021), older persons (de Feijter et al., 2021), or individuals with neurodegenerative conditions (Yang et al., 2023) like Parkinson's or Alzheimer's disease.

In addition, Actigraphy is extensively used to assess sleep and circadian disruptions in populations exposed to irregular schedules, such as shift workers (Baek et al., 2020) and frequent travelers (Feliciano et al., 2019). It helps in studying shift work disorder and the impact of jet lag, which affects circadian alignment. Tracking real-time sleep-wake patterns provides critical insights into the severity of circadian disruptions and the adaptation periods required to recover.

2.1.1 Variables

Since we are working with data from the Condor company, ActTrust, where the reader can find all the same references in this hyperlink: ActTrust References, we have the following variables for each patient:

- **DATE/TIME**: Date and time of the log.
- **MS**: Milliseconds value of the log.
- **EVENT**: Value of the event. Mixed Mode: Indicates whether there was an event during this period.
- **TEMPERATURE**: Value of the temperature (in °C).
- **EXT. TEMPERATURE**: Value of the external temperature (in °C).
- **ORIENTATION**: Value of the device orientation.
- **PIM**: Activity value calculated in Proportional Integral Mode.
- **PIMn**: Value of the PIM normalized per second.
- **TAT**: Activity value calculated in Time Above Threshold mode.

- **TATn**: Value of the TAT normalized per second.
- **ZCM**: Activity value calculated in Zero Crossing Mode.
- **ZCMn**: Value of the ZCM normalized per second.
- **LUX**: Value of the light intensity in lux (Only hardware 2.x).
- **AMB LIGHT**: Value of the ambient light intensity in µw/m² (Only hardware 3.x).
- **RED LIGHT**: Value of red light intensity in µw/cm² (Only hardware 3.x).
- **GREEN LIGHT**: Value of green light intensity in µw/cm² (Only hardware 3.x).
- **BLUE LIGHT**: Value of blue light intensity in µw/cm² (Only hardware 3.x).
- **IR LIGHT**: Value of IR light intensity in µw/cm² (Only hardware 3.x).
- **UVA LIGHT**: Value of Ultraviolet A light intensity in µw/cm² (Only hardware 3.x).
- **UVB LIGHT**: Value of Ultraviolet B light intensity in µw/cm² (Only hardware 3.x).
- **STATE**: Value generated on the sleep score screen that indicates the patient's state: Awake, Sleeping, Resting, OW (device removed), or Editable states.

2.1.2 Metrics

Actigraphy provides several metrics to evaluate various contexts in the field, including M10, L5, and others. These metrics allow researchers to analyze and interpret patterns of movement and rest, offering valuable insights into different aspects of the subject's activity levels. Actigraphy enables a deeper understanding of behavioral and physiological states by measuring parameters like the most active and least active periods. This tool is especially useful in studies on sleep patterns, circadian rhythms, and overall activity monitoring, contributing significantly to the analysis of human behavior and health (Mitchell et al., 2017).

- Most ten ($M10$): represents the 10 hours of highest daily activity, especially for identifying patient activity patterns and cycles.
- Least five ($L5$): the five consecutive hours of lowest average activity.
- Relative Amplitude (RA): the difference between periods of rest and activity defined as follows in Equation 2.1:

$$RA = \frac{M10 - L5}{M10 + L5} \tag{2.1}$$

- Interday Variability (IV): quantifies the alternation between phases of rest and activity. N represents the total number of observations (or measurements) recorded by the actigraph, X_i represents the activity count at the i-th time interval, X_{i-1} represents the activity count at the previous time interval, and X_m represents the mean (average) activity count across all observed time intervals. The value is calculated in Equation 2.2:

$$IV = \frac{N\sum_{i=2}^{N}(X_i - X_{i-1})^2}{(N-1)\sum_{i=1}^{N}(X_m - X_i)^2} \tag{2.2}$$

- Interday Stability (IS): indicates how daylight synchronizes with the patient's activity. N represents the total number of observations (or measurements) recorded by the actigraph over a while, p denotes the number of periods into which the data is divided (often the number of days in the study), X_h represents the average activity count for a specific period h, and X_m represents the overall mean (average) activity count across all observations. The value is calculated in Equation 2.3:

$$IS = \frac{N\sum_{h=1}^{p}(X_h - X_m)^2}{p\sum_{i=1}^{N}(X_m - X_i)^2} \tag{2.3}$$

2.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is a powerful technique for measuring similarity between two time-dependent sequences, commonly called time series. When working with time series, comparing sequences with temporal misalignments or non-linear distortions is often necessary. Two major approaches for this are Euclidean distance and DTW. Euclidean distance provides a straightforward comparison but assumes perfect synchronization between sequences, making it suitable only when the timing between data points is aligned. DTW, however, offers more flexibility by allowing non-linear adjustments to the time axis, making it particularly valuable in applications involving time series data across various domains, such as time series analysis (Salvador and Chan, 2007), speech recognition (Ismail et al., 2020), handwriting recognition, and gesture recognition (Vikram et al., 2013).

The main objective of DTW is to find an optimal alignment between two temporal sequences, say X and Y, by allowing non-linear adjustments along the time axis to minimize their overall distance. Unlike traditional methods, such as Euclidean distance, which assume that the sequences are perfectly synchronized, DTW dynamically stretches and compresses segments of the sequences as needed to obtain the best possible match. This approach is beneficial for data where timing can vary, like speech, where similar phonetic sounds can occur at different speeds, or in motion analysis, where the same gestures may take different lengths to perform.

To achieve this, DTW computes a cumulative cost matrix, which records the cost of aligning each element in sequence X with each component in sequence Y. This cost matrix enables the calculation of the minimum distance path between the sequences by summing up the minor alignment costs. The warping path, a sequence of index pairs, then maps elements from sequence X to elements in sequence Y, indicating the best alignment between them.

DTW's flexibility in handling sequences of varying lengths and timings makes it worthwhile in fields where exact synchronization between time series elements is unrealistic or undesirable. It is also robust to time shifts and distortions, meaning it can capture similarities in patterns that occur at different rates or time frames. This property is essential in real-world applications, where temporal variations often exist due to environmental or biological factors, as in our case.

However, DTW is computationally intensive, especially for long sequences, because it requires the calculation of the entire cumulative cost matrix, which has a complexity of $O(n \times m)$, where n and m are the lengths of the two sequences. This can become a limitation when dealing with large datasets, as DTW can be slower than other distance measures. Additionally, DTW does not directly account for the amplitude differences between sequences, which can be problematic if the comparison requires both timing and amplitude matching. In situations where these problems happen, we may have different techniques to deal with each case, such as fast-DTW and SegmentedDTW to deal with lower computational complexity, Z-normalization, and Derivative DTW to handle normalization problems.

2.3 Normalization and MinMaxScaler

Normalization is a preprocessing technique in ML designed to adjust feature scales so that each feature contributes equally to distance-based calculations. This process is used with algorithms sensitive to feature magnitudes, such as k-nearest neighbors and support vector machines, which rely on accurate distance measurements for optimal performance (Ali, 2022). Generally, normalization has a more pronounced effect on models that leverage distance as a core strategy. In contrast, its impact is less significant in tree-based models like Decision Tree, Random Forest, and XGBoost, which use rule-based paths rather than distances to guide decisions. This way, normalization promotes consistency and enhances learning by rescaling features to a uniform range.

A commonly applied normalization technique transforms each feature individually to a specific range, often between 0 and 1. This scaling is achieved by subtracting the minimum value of each feature and dividing by the range (i.e., the difference between the maximum and minimum values). This linear transformation preserves the relationships between data points while maintaining the original distribution shape. Equation 2.4 defines this normalization formula.

$$x'_i = \frac{x_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \tag{2.4}$$

Where:

- x_i is the original value of the feature for the i-th sample.

- $\min(\mathbf{x})$ is the minimum value of the feature $\mathbf{x}$ in the dataset.
- $\max(\mathbf{x})$ is the maximum value of the feature $\mathbf{x}$ in the dataset.
- x_i' is the normalized value of x_i, scaled to the range [0, 1].

One key advantage of using MinMaxScaler is its simplicity and effectiveness in preparing data for machine learning models that assume normalized input. By aligning feature scales, the MinMaxScaler reduces bias introduced by features with more extensive ranges and facilitates faster convergence in optimization algorithms. This is particularly beneficial for neural networks, where varying feature scales can lead to slower training and convergence issues due to skewed gradient updates.

MinMaxScaler is straightforward to use and integrates seamlessly into a data processing pipeline. Its impact is improving the accuracy and performance of machine learning models across for specific algorithms a wide range of tasks. In summary, normalization via MinMaxScaler is a recommended technique in the data preprocessing arsenal, enabling more robust and reliable model training and deployment across diverse domains.

2.4 Models

Artificial intelligence incorporates a variety of learning methodologies, including supervised, unsupervised, reinforcement learning, and others. Given that the dataset in question consists of time-series data collected over multiple days, with each record precisely timestamped to specific patient events, employing supervised learning—specifically regression analysis—is judiciously choseWe specific algorithms, we will provide illustrative examples to enhance understanding, as the mathematical formalism may not be immediately apparent. All examples will be implemented in the Python programming language, utilizing libraries such as Scikit-learn for machine learning models.

2.4.1 Dummy Regressor

The Dummy method defines possible baselines for the process the user is trying to reach. It uses essential approaches, such as the DummyRegressor, which builds a regressor to give a constant value, the median, or the mean of the analyzed data. It is essential to inform that these methods are not used as final predictors but only for comparison.

In addition to serving as a baseline, the DummyRegressor is valuable for assessing the performance of more complex models by providing a simple reference point. Using basic strategies such as always predicting the mean, median, or constant value allows users to evaluate whether a more sophisticated model adds real value over trivial approaches. The DummyRegressor can also help identify potential data issues. For instance, if a complex model performs similarly to or worse than a DummyRegressor, this might indicate problems such as data noise, poor feature selection, or model overfitting. A running example can be seen in 2.1.

Algorithm 2.1: Dummy Regressor example in Python.

```python
from sklearn.dummy import DummyRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd

# Generate a simple dataset
np.random.seed(42)
X = np.random.rand(100, 1) * 10  # Features
y = 2.5 * X.squeeze() + np.random.randn(100) * 2  # Target with some noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize the Dummy Regressor with 'mean' strategy
dummy_regressor = DummyRegressor(strategy="mean")

# Fit the Dummy Regressor to the training data
dummy_regressor.fit(X_train, y_train)

# Predict the target values for the test set
y_pred = dummy_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
```

2.4.2 Decision Tree Regressor

A Decision Tree algorithm is structured akin to a tree-like data framework, centralizing on interconnected decision nodes via branches that enforce specific data rules to evaluate the assessed statements' veracity. The data is then transmitted to subsequent decision nodes or terminal nodes, commonly referred to as leaves, which do not possess further branching and represent the conclusion of the analysis. In the context of regression tasks, the Decision Tree Regressor scrutinizes the attributes of the problem and, by utilizing predetermined hyperparameters such as maximum depth and number of leaves, forecasts continuous or discrete outcomes.

Decision trees are esteemed for their high interpretability, offering an explicit visual representation of the decision-making process. This attribute renders them particularly advantageous for applications wherein comprehending the model's underlying logic is imperative. Unlike linear models, Decision Trees adeptly manage intricate, non-linear relationships between features and the target variable. Consequently, this capability enhances their robustness across diverse regression tasks. These characteristics highlight the application of decision tree regressors in several areas, from economic problems (Sai et al., 2023) to health care issues (Oyeleye et al., 2022).

Moreover, Decision Trees are scale-invariant, obviating the need for normalizing or scaling features. This characteristic significantly simplifies the preprocessing stage of data analysis. Furthermore, Decision Trees can handle numerical and categorical data, facilitating their application to various problems without requiring extensive data transformations.

Additionally, Decision Trees may undergo pruning during training to circumvent over-fitting. This pruning process diminishes the tree's size by excising segments that contribute minimally to the prediction of the target variable. Lastly, by modifying hyperparameters such as maximum depth, minimum samples split, and minimum samples leaf, users can regulate the model's complexity, balancing under-fitting and over-fitting. A running example can be seen in 2.2.

Algorithm 2.2: Decision Tree Regressor example in Python.

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate a simple dataset
np.random.seed(42)
X = np.sort(np.random.rand(100, 1) * 10, axis=0)  # Features (sorted for
    visualization)
y = np.sin(X).ravel() + np.random.randn(100) * 0.1  # Target with some
    noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize the Decision Tree Regressor
tree_regressor = DecisionTreeRegressor(max_depth=4, random_state=42)

# Fit the regressor to the training data
tree_regressor.fit(X_train, y_train)

# Predict the target values for the test set
y_pred = tree_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean␣Squared␣Error:␣{mse:.2f}")
```

2.4.3 Random Forest Regressor

Random Forest is fundamentally an ensemble learning technique that employs the bagging method. This approach entails constructing multiple Decision Trees, as previously described, each developed from a bootstrap sample of the data, a random selection with replacement from the entire dataset. This methodology aims to reduce the model's variance without significantly elevating bias. By aggregating the predictions of individually cultivated trees through averaging or majority voting methods, Random Forest enhances predictive accuracy and mitigates overfitting, a prevalent issue in single Decision Tree models.

In conjunction with bagging, Random Forest incorporates randomness into selecting features considered for splitting at each node within the decision trees. The algorithm randomly selects a subset of the total features at each node during tree construction. This strategy ensures that the trees remain de-correlated and introduces an additional layer of variance reduction. Typically, this leads to a more robust ensemble that outperforms individual trees when applied to unseen data.

Furthermore, Random Forest features an intrinsic mechanism to estimate its prediction error without requiring a separate validation dataset, known as the out-of-bag (OOB) error estimation. The algorithm constructs each tree in the forest from a distinct bootstrap sample drawn from the original dataset. The data points excluded from this sample, referred to as the OOB data, serve to assess the performance of each tree. The OOB error, calculated as the average error of predictions on the OOB samples across all trees, is an unbiased estimator of the model's overall performance.

Lastly, Random Forest is particularly adept in scenarios involving large datasets (Zainab et al., 2020) with numerous features, as in our situation. It can manage thousands of input variables without necessitating feature elimination and efficiently addresses missing data. Furthermore, the ensemble approach employed by Random Forest generally results in enhanced performance and greater robustness than a single decision tree can achieve. A running example can be seen in 2.3.

Algorithm 2.3: Random Forest Regressor example in Python.

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate a simple dataset
np.random.seed(42)
X = np.sort(np.random.rand(100, 1) * 10, axis=0)  # Features (sorted for
    visualization)
y = np.sin(X).ravel() + np.random.randn(100) * 0.1  # Target with some
    noise

```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize the Random Forest Regressor
forest_regressor = RandomForestRegressor(n_estimators=100, max_depth=4,
    random_state=42)

# Fit the regressor to the training data
forest_regressor.fit(X_train, y_train)

# Predict the target values for the test set
y_pred = forest_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean␣Squared␣Error:␣{mse:.2f}")
```

2.4.4 XGBoost Regressor

XGBoost is one of the many gradient boosting frameworks, such as LightGBM [1] and CatBoost [2]. The gradient boosting process also comes from tree variation algorithms, such as random forest. Still, instead of planting a load of independent trees at random (the bagging technique we explained before), each tree created will be weighted to compensate for any residual error in the previous tree, along with the hyperparameter of max_depth, $n\_estimators$ and now with a $learning\_rate$ hyperparameter (where its range is between 0 and 1).

Also, XGBoost includes L1 (Lasso Regression) and L2 (Ridge Regression) regularization terms in its cost function, which are not typical in standard implementations of gradient boosting that we cited before. This regularization helps reduce the overfitting process, making XGBoost particularly robust against model complexity, even when dealing with many features and datasets with multiple dimensions.

Moreover, we can use XGBoost for regression, classification (binary and multi-class), ranking, and user-defined prediction scenarios. It supports user-defined objective functions and evaluation criteria, adding a layer of flexibility for various scientific and business problems. With such an enormous flexibility, as many tree-based models have, XGBoost can be applied in different scenarios, such as health (Jyothsna et al., 2022), and education (Jeganathan et al., 2022). A running example can be seen in 2.4.

Algorithm 2.4: XGBoost Regressor example in Python.

```
from xgboost import XGBRegressor
```

[1] https://www.microsoft.com/en-us/research/project/lightgbm/
[2] https://catboost.ai/

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate a simple dataset
np.random.seed(42)
X = np.random.rand(100, 1) * 10  # Features
y = np.sin(X).ravel() + np.random.randn(100) * 0.1  # Target with some
    noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize the XGBoost Regressor
xgb_regressor = XGBRegressor(n_estimators=100, max_depth=4, learning_rate
    =0.1, random_state=42)

# Fit the regressor to the training data
xgb_regressor.fit(X_train, y_train)

# Predict the target values for the test set
y_pred = xgb_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean␣Squared␣Error:␣{mse:.2f}")
```

2.4.5 Support Vector Regressor

The Support Vector Regressor (SVR) is a variation of the Support Vector Machine (SVM), adapted for regression problems instead of classification. SVR aims to fit a function approximating the target values within a tolerance margin (ϵ). We use the ϵ-insensitive loss function, so we do not penalize errors within the margin ϵ. The SVR algorithm identifies the best line or hyperplane that fits the data while ensuring most data points fall within this margin. Only the data points outside this margin, known as support vectors, influence the model. The objective of SVR is to minimize the complexity of the model (i.e., the norm of the weights) while penalizing any deviations from the margin that exceeds ϵ, thereby achieving a balance between model complexity and prediction accuracy.

Furthermore, one of the critical strengths of SVR, similar to SVM for classification, is its ability to handle non-linear relationships in the data using the kernel trick. Kernels such as polynomial, radial basis function (RBF), and sigmoid can be applied to transform the input features into higher-dimensional spaces where a linear regression model can be more effective.

Additionally, we need to tune several hyperparameters in SVR for optimal performance. These include the regularization parameter C, which controls the trade-off between achieving a low error on the training data and minimizing the model complexity. This ϵ parameter defines the width of the insensitive zone and parameters associated with the chosen kernel (e.g., the gamma parameter for the RBF kernel).

Furthermore, we can apply SVR in various scenarios, such as predicting environmental disasters (Asim et al., 2018), financial accounting (Chakri et al., 2023), geology (Zhu et al., 2018), and others. A running example can be seen in 2.5.

Algorithm 2.5: Support Vector Regressor example in Python.

```python
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate a simple dataset
np.random.seed(42)
X = np.random.rand(100, 1) * 10  # Features
y = np.sin(X).ravel() + np.random.randn(100) * 0.1  # Target with some
    noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Standardize the features (important for SVR)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize the Support Vector Regressor
svr_regressor = SVR(kernel="rbf", C=1.0, epsilon=0.1)

# Fit the regressor to the training data
svr_regressor.fit(X_train_scaled, y_train)

# Predict the target values for the test set
y_pred = svr_regressor.predict(X_test_scaled)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean␣Squared␣Error:␣{mse:.2f}")
```

2.4.6 K-Nearest Neighbors Regressor

The K-Nearest Neighbors (KNN) algorithm is a non-parametric method used for both classification and regression tasks. In numerical regression, KNN makes predictions by identifying the K nearest data points (neighbors) to the given input based on a specified distance metric (commonly Euclidean distance). The algorithm then predicts the target value for the input by computing the average of the target values of these K nearest neighbors. This method leverages the intuition that data points with similar features will also have identical target values, producing predictions that reflect local patterns in the dataset.

The KNN regression relies heavily on the choice of distance metric to identify the nearest neighbors. While we commonly use Euclidean distance, we can also employ other distance measures, such as Manhattan, Minkowski, or cosine distance, depending on the nature of the data and the problem at hand.

Moreover, the performance of KNN regression depends on selecting the parameter K, which determines the number of nearest neighbors to consider. A small value of K makes the model sensitive to noise (over-fitting), while a significant value of K can smooth out the predictions too much (under-fitting). We typically use cross-validation to choose an optimal K. One of the many techniques to find the ideal K is the "Elbow Technique," where he can select a series of K-values from a defined range and look at the result of each K, choosing the best K in the end. KNN is another famous algorithm that we can apply in areas like energy prediction (Khan and Byun, 2020), healthcare problems (Niemeijer et al., 2008), and physics (Durbin et al., 2021). A running example can be seen in 2.6.

Algorithm 2.6: K-Nearest Neighbors Regressor example in Python.

```python
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate a simple dataset
np.random.seed(42)
X = np.random.rand(100, 1) * 10  # Features
y = np.sin(X).ravel() + np.random.randn(100) * 0.1  # Target with some
    noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize the KNN Regressor
knn_regressor = KNeighborsRegressor(n_neighbors=5, weights='uniform')

# Fit the regressor to the training data
```

```
knn_regressor.fit(X_train, y_train)

# Predict the target values for the test set
y_pred = knn_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean␣Squared␣Error:␣{mse:.2f}")
```

2.4.7 Linear Regressor

The Linear Regression that we are using in this context is the Ordinary Least Squares (OLS) method for Linear Regression, which aims to fit a linear model by determining the coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ that minimize the residual sum of squares between the observed target values in the dataset and the target values predicted by the linear model. This approach seeks to find the best-fitting line (or hyperplane in higher dimensions) that reduces the discrepancy between the actual and predicted values, thereby providing the most accurate approximation of the underlying relationship between the dependent and independent variables.

Linear regression relies on several key assumptions. First, the assumption of linearity posits that the relationship between the dependent and independent variables is linear. Second, the independence assumption requires that each observation is independent of the others. Third, homoscedasticity implies that the variance of the error terms remains constant across all levels of the independent variables. Finally, the normality assumption asserts that the error terms follow a normal distribution.

Linear Regression, as a classic regression structure, has been applied in the most different areas, such as health (Nichols et al., 2022), politics (Han et al., 2023), and even in energy consumption (Olu-Ajayi et al., 2022). A running example can be seen in 2.7.

Algorithm 2.7: Linear Regressor example in Python.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate a simple dataset
np.random.seed(42)
X = np.random.rand(100, 1) * 10  # Features
y = 2.5 * X.squeeze() + np.random.randn(100) * 2  # Target with some noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
```

```
# Initialize the Linear Regressor
linear_regressor = LinearRegression()

# Fit the regressor to the training data
linear_regressor.fit(X_train, y_train)

# Predict the target values for the test set
y_pred = linear_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean␣Squared␣Error:␣{mse:.2f}")

# Display model coefficients
print(f"Coefficient:␣{linear_regressor.coef_[0]}")
print(f"Intercept:␣{linear_regressor.intercept_}")
```

2.5 Autoencoder

Although autoencoders are AI models, we have dedicated a separate section to them, as they follow a unique structure within Deep Learning (DL) and differ from traditional models. Autoencoders are primarily used for unsupervised learning to encode data efficiently. Designed to learn a compressed representation, or encoding, of a dataset, autoencoders are often applied for tasks such as dimensionality reduction or feature extraction. Their basic structure consists of three main components: an encoder, which compresses the data; a latent space, which holds the compact representation; and a decoder, which reconstructs the input data. Autoencoders are widely applied in areas like image denoising (Tripathi, 2021), dimensionality reduction (Lin et al., 2020), anomaly detection (Torabi et al., 2023), and feature extraction (Yu et al., 2021), where they reveal hidden patterns and enhance computational efficiency by working with lower-dimensional representations. Advanced variations, such as convolutional autoencoders (Bedi and Gole, 2021), variational autoencoders (Kim et al., 2021) and Transformer Masked Autoencoder (Mendonça Freire and Curi, 2024), further expand their functionality, making them powerful tools for both generative tasks and complex data processing.

2.5.1 General Autoencoder Structure

The architecture of an autoencoder consists of three key components that are described in the following items and can also be seen in the Figure 2.1.

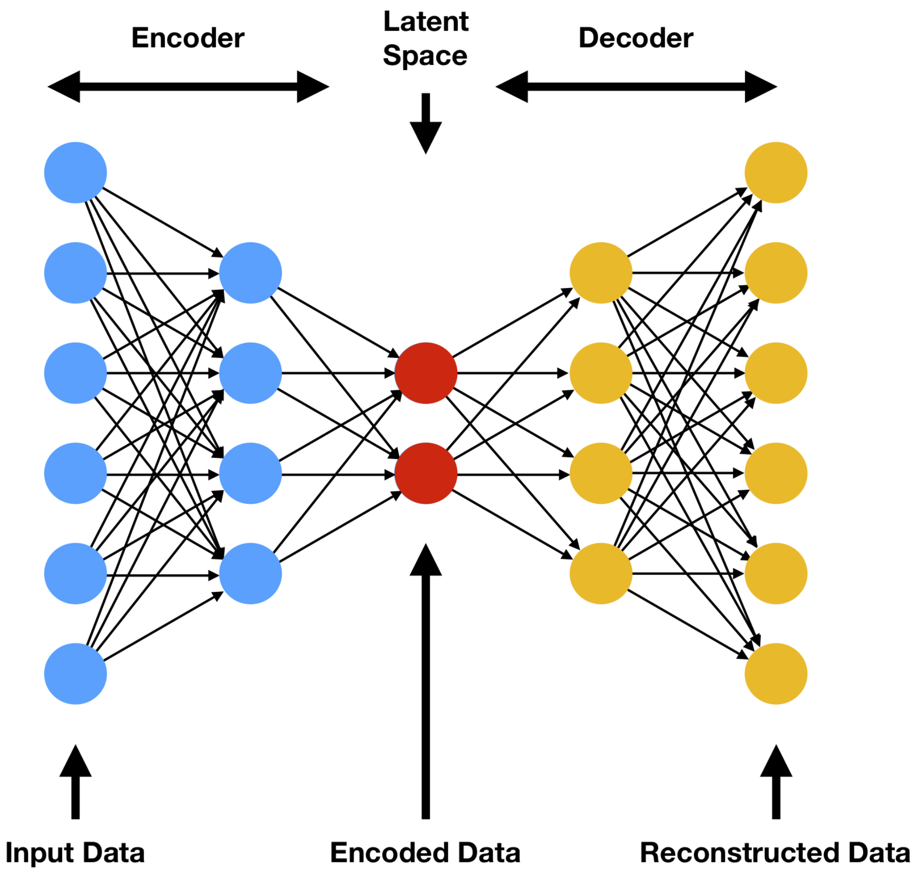


Figure 2.1: Autoencoder structure.

- **Encoder**: The encoder compresses the input data into a latent representation, reducing its dimensionality and encoding it into a more compact form. Mathematically, the encoder can be represented as a function $E(x) = z$, where x denotes the input data and z represents the encoded form. This process typically involves linear transformations, such as $z = W \cdot x + b$, where W and b are the weight matrix and bias vector, followed by non-linear activation functions like ReLU, sigmoid, or $\tanh$. In specific models, such as transformers, an attention mechanism is integrated to help the encoder focus on the most relevant parts of the input. The resulting output z is a reduced-dimensional representation optimized to minimize a loss function that captures the essential structure of the input data.

- **Latent Space**: The latent space is the compact, encoded representation of the input data, containing the essential features necessary to reconstruct the original input. The latent space is typically of a lower dimension than the input space, forcing the network to learn efficient representations that are both meaningful and minimal.

- **Decoder**: The decoder reconstructs the input data from the encoded latent representation. Conceptually, the decoder can be viewed as a function $D(z) = x'$, where x' is the reconstructed version of the input. The decoding process applies a transformation sequence that maps the latent representation z back to the original input space. These transformations may include linear operations, such as $x' = W' \cdot z + b'$, where W' and b' are the weight matrix and bias vector specific to the decoder, followed by non-linear activation functions to enhance reconstruction quality. In some models, the decoder also incorporates attention mechanisms to focus on relevant portions of the encoded representation. The final output x' aims to approximate the original input x closely, and the decoder is trained to minimize a loss function measuring the difference between x and x', ensuring that key features of the input data are retained.

2.5.2 Advantages and Disadvantages of Autoencoders:

Autoencoders have several strengths and limitations, particularly relevant when dealing with complex data such as actigraphy.

- **Advantages:**

- **Handling Non-Linear Relationships**: Autoencoders can capture complex, non-linear patterns, making them highly suitable for modeling intricate structures in actigraphy data.
- **Dimensionality Reduction**: Autoencoders achieve dimensionality reduction by learning compressed representations, minimizing noise, and focusing on features essential for accurate imputation.
- **Flexibility**: They can be customized for different data types and optimized for specific imputation tasks, offering flexibility in handling various missing data scenarios.
- **Scalability**: Autoencoders efficiently handle large datasets, making them ideal for extensive actigraphy datasets collected over extended periods.
- **Automatic Feature Learning**: Autoencoders learn features automatically from the data without requiring extensive manual feature engineering, saving time and reducing the need for domain-specific expertise.

- **Disadvantages:**

- **Complexity**: Training autoencoders can be computationally intensive and time-consuming, particularly for large actigraphy datasets, requiring significant resources.
- **Data Requirements**: Autoencoders require large amounts of data to train effectively, and performance may decline if the actigraphy dataset is limited.
- **Overfitting**: There is a risk of overfitting, where the autoencoder may learn noise rather than meaningful patterns, leading to poor generalization on unseen data.

- **Hyperparameter Tuning**: Autoencoder performance depends heavily on the choice of hyperparameters, such as the number of layers, neurons, and learning rate, which can be challenging and time-intensive to optimize.
- **Interpretability**: The internal representations learned by autoencoders are often difficult to interpret, complicating understanding how the model imputes missing data.
- **Imputation Bias**: If the missing data pattern is non-random, the autoencoder may introduce bias in the imputation, potentially affecting subsequent analyses.

2.5.3 Transformed Masked Autoencoder

The Transformed Masked Autoencoder (TMAE) is an advanced variation of the traditional autoencoder designed to enhance the handling of missing data by incorporating a masking mechanism. In this approach, sections of the input data are masked during training, simulating missing data patterns. The autoencoder is then trained to reconstruct the complete data from these partial inputs, thereby learning to handle gaps in the data effectively.

- **Masking Mechanism**: The TMAE applies a masking operation to random sections of the input data, effectively hiding certain values. This masking allows the autoencoder to be trained in a way that mimics real-world scenarios with incomplete data. By learning to reconstruct missing segments, the TMAE is better equipped for imputation tasks, as it has been explicitly trained to handle incomplete data.
- **Transformation Layer**: Besides masking, TMAEs often include a transformation layer that processes the masked input, helping to reformat it in a way that preserves essential data patterns. This transformation improves the model's ability to detect and replicate underlying patterns, even in gaps. For instance, positional encodings or learned embeddings may retain context and temporal structure within the data.
- **Reconstruction Process**: The decoder in a TMAE reconstructs the missing data by predicting the masked values based on the patterns learned during training. This process enables the model to fill in missing values and capture temporal dependencies and correlations within the data, making it particularly effective for actigraphy datasets with irregular missing patterns.

The TMAE's design makes it especially suitable for tasks where missing data is standard, as it has been optimized to reconstruct data with partial visibility. This model adapts well to actigraphy applications, where data collection inconsistencies, such as off-wrist episodes, can result in gaps. By explicitly training on masked data, the TMAE provides a more robust and reliable solution for imputing missing values, capturing both linear and non-linear relationships in the data.

2.6 Hyperparameter Optimization

Optimization plays a vital role in AI because the performance of algorithms is significantly affected by the choice of hyperparameters. While each algorithm has a default set of hyperparameters, relying solely on these defaults is often insufficient, as they may not yield optimal results for different problems. Consequently, exploring various hyperparameter combinations is essential, considering available options, search time complexity, and resource consumption.

2.6.1 Grid Search

Grid Search is an exhaustive method for hyperparameter optimization, systematically evaluating every specified combination for the analyzed model. By default, Grid Search employs cross-validation, which divides the dataset into multiple parts, training and testing the AI model with different dataset splits. It ultimately selects the best configuration based on the desired metric. In this way, we can resume the steps to:

- Defining the hyperparameters and their ranges.
- Creating a grid of all possible hyperparameter combinations.
- Training and evaluating the model for each combination using cross-validation.
- Selecting the best hyperparameters based on performance.

2.6.2 Randomized Search

Randomized Search can be a more efficient alternative to Grid Search, mainly when dealing with an ample hyperparameter space. Grid Search employs a discrete approach, evaluating every possible combination of specified values, which can be time-consuming and resource-intensive. In contrast, a Randomized Search randomly selects hyperparameters from the specified values, continuing until it reaches a predefined number of iterations. The primary steps involved include:

- Defining the hyperparameters and their distributions.
- Specifying the number of iterations (random samples).
- Randomly sampling hyperparameter combinations.
- Training and evaluating the model for each sampled combination using cross-validation.
- Selecting the best hyperparameters based on performance.

2.7 Metrics

To evaluate each regressor that we are using to perform data imputation in each patient dataset, we need statistical metrics to do so. In this context, we selected five metrics to do so, which are R^2, MSE, RMSE, MAE, and MASE.

2.7.1 Coefficient of Determination

The coefficient of determination, also known as R², measures the proportion of variability in the dependent variable explained by the model. It is defined by the Equation 2.5.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.5}$$

where y_i is the actual value of the dependent variable; $\hat{y}_i$ is the predicted value of the dependent variable; $\bar{y}$ is the mean of the actual values y_i; and n is the number of observations.

The R^2 value ranges from 0 to 1, where $R^2 = 1$ indicates that the regression model perfectly fits the data and $R^2 = 0$ suggests that the model does not explain any of the variability in the response variable around its mean. Higher values of R^2 indicate a better fit of the model.

2.7.2 Mean Squared Error

The Mean Squared Error (MSE) is a standard measure of the average of the squares of the errors. We define it in the Equation 2.6.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.6}$$

where y_i is the actual value of the dependent variable; $\hat{y}_i$ is the predicted value of the dependent variable and; n is the number of observations.

The MSE measures the average of the squares of the errors, which are the differences between the actual and predicted values. Lower MSE values indicate a better fit of the model to the data.

2.7.3 Root Mean Squared Error

The Root Mean Squared Error (RMSE) measures the differences between values predicted by a model and the values observed. It is the square root of the average of the squared differences. We can define it as shown in Equation 2.7.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.7}$$

where y_i is the actual value of the dependent variable; $\hat{y}_i$ is the predicted value of the dependent variable and; n is the number of observations.

The RMSE quantifies the magnitude of the error. Lower values of the RMSE indicate a better fit of the model to the data.

2.7.4 Mean Absolute Error

The Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon. We can define it as shown in Equation 2.8.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.8}$$

where y_i is the actual value of the dependent variable; $\hat{y}_i$ is the predicted value of the dependent variable and; n is the number of observations.

The MAE measures the average magnitude of the errors in a set of predictions without considering their direction. It is the average absolute difference between prediction and actual observation over the test sample. Lower values of MAE indicate a better fit of the model to the data.

2.7.5 Mean Absolute Scaled Error

The Mean Absolute Scaled Error (MASE) measures the accuracy of a forecast model. We can define it as shown in Equation 2.9.

$$\text{MASE} = \frac{\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\frac{1}{n-1}\sum_{i=2}^{n} |y_i - y_{i-1}|} \tag{2.9}$$

where y_i is the actual value of the dependent variable; $\hat{y}_i$ is the predicted value of the dependent variable; n is the number of observations and; y_{i-1} is the actual value at the previous time step.

The MASE scales the mean absolute error by the in-sample mean absolute error from a naive forecast model. It is scale-independent and can compare forecast accuracy across various scales. Lower values of MASE indicate a better fit of the model to the data.

2.8 SHapley Additive exPlanations (SHAP)

SHAP, or SHapley Additive exPlanations, is an interpretation technique for AI models that helps explain the predictions of complex models, including regression models, by assigning each feature an importance value for a given prediction. Based on cooperative game theory, SHAP provides a method to quantify each feature's contribution to the prediction outcome fairly and consistently. SHAP is widely used in different areas, such as Natural Language Processing

(NLP) (Mosca et al., 2022), autoencoders (Antwarg et al., 2021), and ML models (Nohara et al., 2022).

In regression models, where the goal is to predict a continuous outcome (e.g., activity levels, temperature, or time spent in specific behaviors), SHAP can provide valuable insights into how each feature influences the predicted value. This information is beneficial for:

- **Understanding Feature Importance**: SHAP identifies which features influence the model's predictions most. For example, in a model predicting daily activity levels, SHAP values might show if variables like light intensity or sleep patterns significantly impact the prediction.
- **Quantifying Feature Contributions**: Each SHAP value represents the contribution of a particular feature to the model's prediction for a specific instance. For example, if a regression model predicts a patient's activity level as 75, SHAP values could reveal that light intensity contributed +10 to this prediction. In contrast, temperature contributed -5, explaining how each feature influenced this outcome.

SHAP values are calculated by evaluating the impact of each feature across different combinations of feature subsets, resulting in an "average" impact score. This process can be summarized in the following steps:

1. **Baseline Prediction**: SHAP starts with a baseline prediction, often the average prediction across all instances in the dataset.
2. **Feature Impact Calculation**: SHAP calculates how including that feature changes the prediction relative to the baseline for each feature. This is done by considering all possible combinations of other features, ensuring that SHAP values are fair and consistent, akin to Shapley values in game theory.
3. **Summing Contributions**: The final prediction is obtained by adding up the SHAP values of each feature plus the baseline prediction. Thus, each feature's SHAP value explains its contribution to the final prediction.

SHAP values are useful for understanding both global and local feature importance in regression models:

- **Global Interpretability**: By averaging SHAP values across all instances in the dataset, we gain insight into which features have the highest overall impact on predictions. This is useful for identifying the importance of features in regression models, helping to determine which variables the model relies on most consistently.

- **Local Interpretability**: SHAP values for individual predictions provide instance-specific explanations, showing how each feature influenced a particular prediction. For example, SHAP can reveal which features significantly affected a specific patient's predicted activity level, providing detailed insights into the model's behavior for each case.

Overall, SHAP enhances the interpretability of regression models by clearly separating feature contributions to each prediction. This transparency is crucial for validating model behavior and understanding how specific features drive the model's predictions.

2.8.1 Related Works

The work developed by Tonon et al. (2022) presents a method for handling OW episodes in actigraphy time series of motor activity, using data from two records (one with a regular sleep-wake cycle and another with an irregular cycle) collected over 14 consecutive days with 1-minute sampling intervals. To simulate real-life OW episodes, artificial missing value (NA) intervals of various durations of hours (1, 2, 4, 6, 12, and 24) and random NA episodes following probabilistic rules were introduced into the data. These missing episodes were replaced with different imputation methods: zeros (representing immobility), the mean, or the median of the corresponding time points from the remaining 13 days. The results showed that missing episodes up to 12 hours caused less than a 5% deviation from the original values. The study recommends avoiding zero imputation when possible due to the risk of artificially reducing variability. Instead, it suggests treating zeros as NAs to prevent inflating invariance. If specific parameters cannot be calculated with NAs present, using the weekly mean of corresponding time points is recommended as an alternative approach. This study brings great ideas and analysis, indicating the scenarios where it is worth performing data imputation or not in actigraphy data. Although, it only uses classical statistical methods in the imputation phase.

The work developed by Jang et al. (2020) explores a DL approach for imputing missing values in actigraphy data, utilizing a denoising convolutional autoencoder trained on accelerometer-based activity records from the National Health and Nutrition Examination Survey. The model was validated using external datasets, including daily activity counts from the Korea National Health and Nutrition Examination Survey and a chronic cerebrovascular disease biobank. The evaluation involved introducing artificial missing data and comparing the DL model's imputation performance against traditional methods like mean imputation, zero-inflated Poisson regression, and Bayesian regression. Results indicated that the DL approach achieved lower partial root mean square error (RMSE) and partial mean absolute error (MAE) than the conventional methods, demonstrating superior accuracy in reconstructing missing values. The study highlights the potential of DL techniques for handling incomplete actigraphy data, suggesting that such methods may outperform traditional approaches by learning complex data patterns without relying on predefined statistical assumptions. Although the Autoencoder outperforms the other methods, its performance still shows relatively high metrics, such as an

RMSE of approximately 800. Since a lower RMSE indicates better accuracy, ideally closer to zero, this result suggests room for further improvement.

The authors of Weed et al. (2022) aims to assess how the timing and duration of missing data in actigraphy records affect the calculation of internal stability (IS) and intraday variability (IV). Additionally, it evaluates the effectiveness of three imputation methods—linear interpolation, mean time of day (ToD), and median ToD imputation—in estimating these metrics. To conduct the analysis, week-long actigraphy datasets with no original data gaps were systematically masked with zeros or 'Not a Number' (NaN) values across different timings and durations, simulating single and multiple missing data episodes. IV and IS calculations were performed on the original (true), masked, and imputed datasets. Results showed that simulated missing data led to significant deviations in IV and IS, particularly for longer gaps, around midday, and when missing periods occurred at similar times across consecutive days. Among the tested methods, median ToD imputation resulted in the most minor deviations, making it the most reliable approach for estimating IV and IS when handling missing data durations of less than 24 hours.

Consequently, the study recommends using median ToD imputation to best preserve the integrity of IS and IV calculations under these conditions. The study only evaluates three basic imputation methods, which limits its scope. More advanced techniques, such as machine learning-based imputation (e.g., Random Forest or autoencoders), could have been included to provide a broader assessment of available approaches.

The classification process presented in Haghayegh et al. (2020) shows the impact of different modes of operation and the potential benefits of a DL approach to evaluate the estimation of sleep parameters using wrist actigraphy. The study aimed to answer two key questions: (1) Does the mode of operation—Proportional Integrating Measure (PIM), which measures movement intensity, or Zero Crossing Mode (ZCM), which measures movement frequency—affect sleep scoring accuracy? (2) Can a DL model that combines PIM and ZCM data outperform existing IAs? The research involved nighttime sleep data from 40 healthy adults, including ZCM and PIM actigraphy data, alongside electroencephalographic (EEG) recordings as the reference. The initial analysis applied several classic DL models to PIM-only, ZCM-only, and combined PIM/ZCM data. Results indicated that the combined ZCM/PIM mode produced a higher agreement with the EEG reference for sleep/wake scoring than ZCM or PIM alone. The new DL model achieved 87.7% accuracy, 94.1% sensitivity, 64.0% specificity, and 59.9% Kappa agreement, showing improvements over other IAs, particularly in specificity and Kappa agreement. The model's estimates for sleep onset latency (SOL), wake after sleep onset (WASO), total sleep time (TST), and sleep efficiency (SE) did not significantly differ from the EEG reference, with less bias and more minor detectable changes than the other IAs. While the DL model shows high accuracy and sensitivity, the specificity (64.0%) and Kappa agreement (59.9%) are relatively low, indicating potential challenges in distinguishing between sleep and wake states.

The works summarized above provide foundational methods for addressing missing data in actigraphy studies and have advanced data imputation techniques. However, these studies often rely on traditional statistical approaches, limited sample sizes, or simplified models that restrict their ability to generalize across diverse actigraphy datasets. Classical statistical methods, while helpful, may lack the sophistication needed to capture the nuanced patterns in complex datasets, particularly for imputation during OW intervals. Similarly, some DL models, although more accurate than conventional methods, display limitations in specific metrics, such as root mean square error (RMSE) and specificity, indicating room for further improvement. The small sample sizes in several studies also hinder the models' applicability to broader populations, potentially reducing their effectiveness in diverse real-world scenarios. Furthermore, the limited exploration of hybrid and patient-specific models suggests an opportunity to improve current methods by incorporating advanced deep learning frameworks. These enhancements could lead to more robust and reliable imputation techniques, ultimately supporting more accurate patient activity and sleep pattern monitoring.

2.9 Final Considerations

In this chapter, we have laid the groundwork for understanding the theoretical elements required for our data imputation methodology. We began by examining critical concepts in actigraphy, emphasizing the significance of variables, such as activity and light metrics, and specific actigraphy measures, like M10 and L5, used to assess circadian rhythms and activity levels. Understanding these elements is essential, as they form the basis of the imputation process in datasets with missing values due to off-wrist (OW) moments.

Next, we discussed Dynamic Time Warping (DTW) and Euclidean distance, two critical techniques for comparing time-dependent sequences. While Euclidean distance offers a straightforward approach, it is only effective when the time sequences are aligned. DTW, however, is a flexible tool that allows non-linear alignment adjustments, making it suitable for analyzing time series data with temporal distortions, such as those found in actigraphy. This capability is precious when comparing periods of activity that may vary in duration across days or individuals, a common scenario in actigraphy-based studies.

We also introduced various machine learning models, mainly focusing on regressors. Tree-based models, including Decision Trees, Random Forests, and XGBoost, offer robust predictive capabilities by constructing ensembles or combining multiple decision paths. Distance-based models like Support Vector Regression (SVR) and K-Nearest Neighbors (KNN) leverage data point proximity, while linear regression provides a more straightforward, interpretable approach. Each model type brings unique advantages for handling actigraphy data, especially when dealing with gaps due to OW periods, where imputing missing values accurately would be meaningful for analysis.

Further, we covered autoencoders—a deep learning approach that compresses and reconstructs data through encoding and decoding processes. This approach is advantageous for complex imputation tasks, allowing us to capture non-linear relationships within the data, especially beneficial for handling extensive and intricate actigraphy datasets.

Additionally, we reviewed essential data preprocessing steps, including normalization techniques like MinMaxScaler, and outlined hyperparameter optimization strategies, such as Grid Search and Randomized Search, to fine-tune model performance. Finally, we discussed the metrics, including R^2, MSE, RMSE, MAE, and MASE, that will be used to evaluate the effectiveness of our models in imputing missing data.

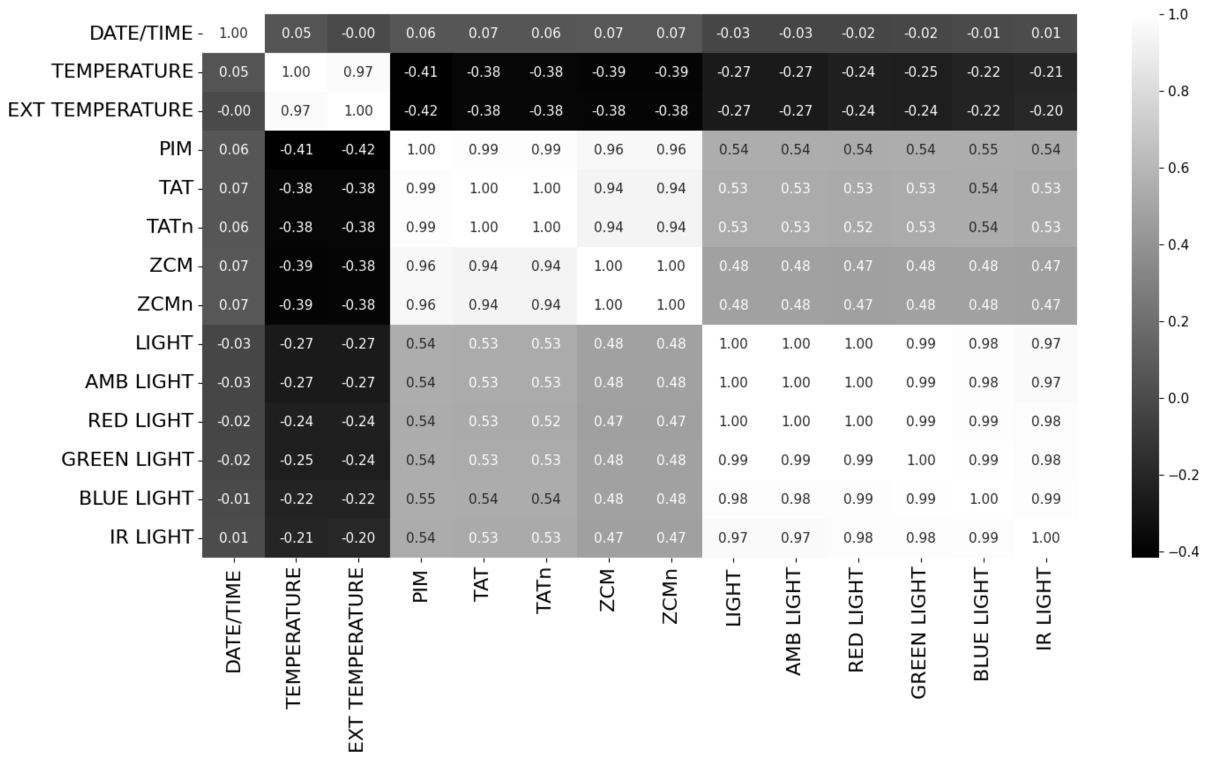
Overall, this theoretical foundation provides a comprehensive understanding of the tools and techniques we will use to develop a robust data imputation pipeline for actigraphy data in the upcoming sections.

Chapter 3

Methodology

Before detailing each step of the proposed methodology, we begin with an Exploratory Data Analysis (EDA), which provided valuable insights that guided several key decisions, such as identifying columns to drop and gaining a clearer understanding of the data structure. The dataset contains records from 23 patients collected between 2018 and 2021 using the ActTrust device developed by Condor. Each record includes multiple parameters that comprehensively view each patient's activities and conditions. Section 2, specifically subsection 2.1.1, provides a complete description of these variables for each patient.

Figure 3.1 presents the correlation values between all the variables in each dataset. This figure provides insight into the relationships and interactions among the various parameters, aiding in the analysis and interpretation of the data. Also, Figures 3.2 and 3.3 show distribution for both PIM and Temperature values, and Figure 3.4 shows the boxplots for ”Temperature x PIM” indicating the interquartile range (IQR) for temperature within each PIM value category. The horizontal line inside the box represents the median temperature. The whiskers extend to the rest of the distribution, except for the outliers, plotted individually using dots.



	DATE/TIME	TEMPERATURE	EXT TEMPERATURE	PIM	TAT	TATn	ZCM	ZCMn	LIGHT	AMB LIGHT	RED LIGHT	GREEN LIGHT	BLUE LIGHT	IR LIGHT
DATE/TIME	1.00	0.05	-0.00	0.06	0.07	0.06	0.07	0.07	-0.03	-0.03	-0.02	-0.02	-0.01	0.01
TEMPERATURE	0.05	1.00	0.97	-0.41	-0.38	-0.38	-0.39	-0.39	-0.27	-0.27	-0.24	-0.25	-0.22	-0.21
EXT TEMPERATURE	-0.00	0.97	1.00	-0.42	-0.38	-0.38	-0.38	-0.38	-0.27	-0.27	-0.24	-0.24	-0.22	-0.20
PIM	0.06	-0.41	-0.42	1.00	0.99	0.99	0.96	0.96	0.54	0.54	0.54	0.54	0.55	0.54
TAT	0.07	-0.38	-0.38	0.99	1.00	1.00	0.94	0.94	0.53	0.53	0.53	0.53	0.54	0.53
TATn	0.06	-0.38	-0.38	0.99	1.00	1.00	0.94	0.94	0.53	0.53	0.52	0.53	0.54	0.53
ZCM	0.07	-0.39	-0.38	0.96	0.94	0.94	1.00	1.00	0.48	0.48	0.47	0.48	0.48	0.47
ZCMn	0.07	-0.39	-0.38	0.96	0.94	0.94	1.00	1.00	0.48	0.48	0.47	0.48	0.48	0.47
LIGHT	-0.03	-0.27	-0.27	0.54	0.53	0.53	0.48	0.48	1.00	1.00	1.00	0.99	0.98	0.97
AMB LIGHT	-0.03	-0.27	-0.27	0.54	0.53	0.53	0.48	0.48	1.00	1.00	1.00	0.99	0.98	0.97
RED LIGHT	-0.02	-0.24	-0.24	0.54	0.53	0.52	0.47	0.47	1.00	1.00	1.00	0.99	0.99	0.98
GREEN LIGHT	-0.02	-0.25	-0.24	0.54	0.53	0.53	0.48	0.48	0.99	0.99	0.99	1.00	0.99	0.98
BLUE LIGHT	-0.01	-0.22	-0.22	0.55	0.54	0.54	0.48	0.48	0.98	0.98	0.99	0.99	1.00	0.99
IR LIGHT	0.01	-0.21	-0.20	0.54	0.53	0.53	0.47	0.47	0.97	0.97	0.98	0.98	0.99	1.00

Figure 3.1: Correlation Matrix - Patient 04.

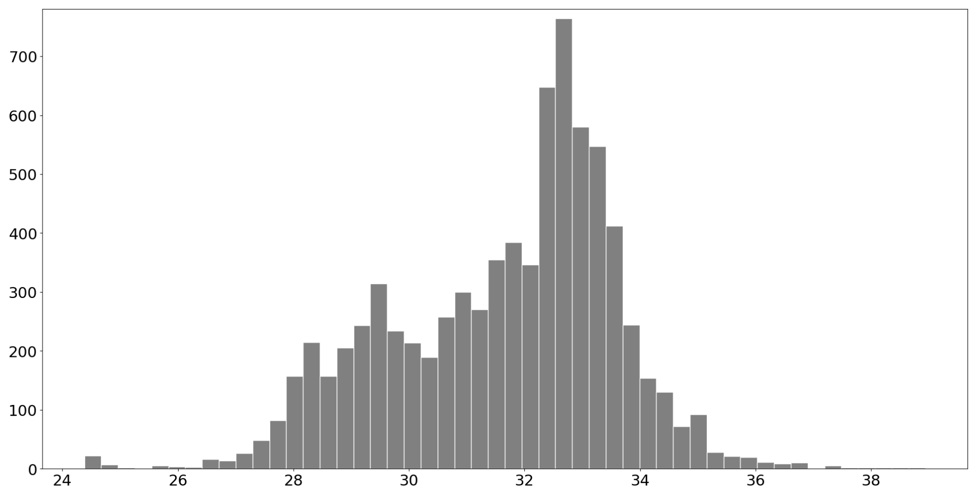


Figure 3.2: Temperature distribution - patient 04.

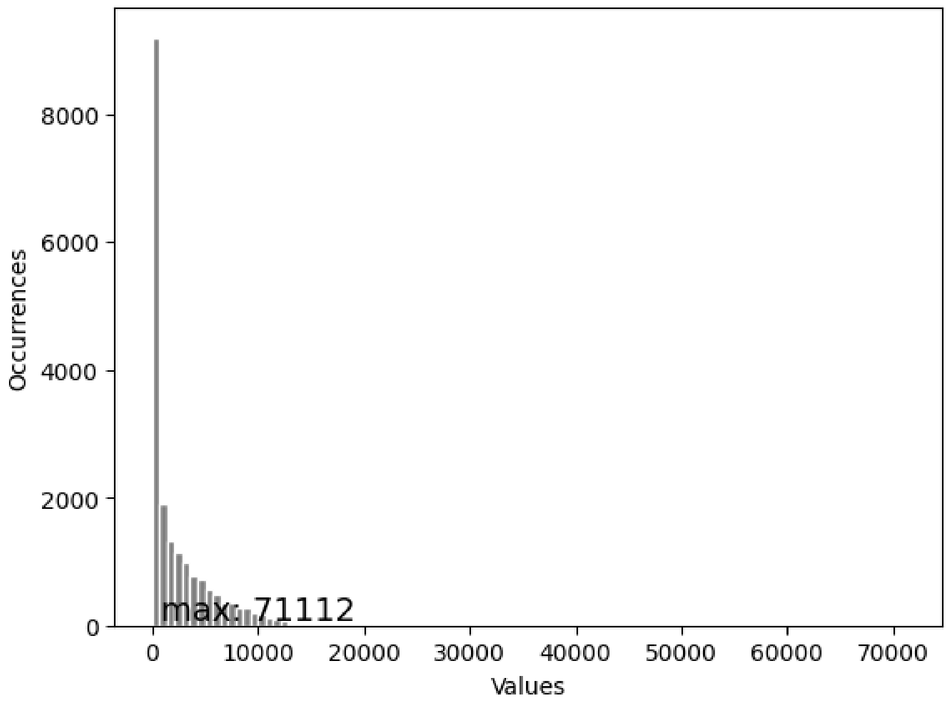


Figure 3.3: PIM distribution - Patient 04.

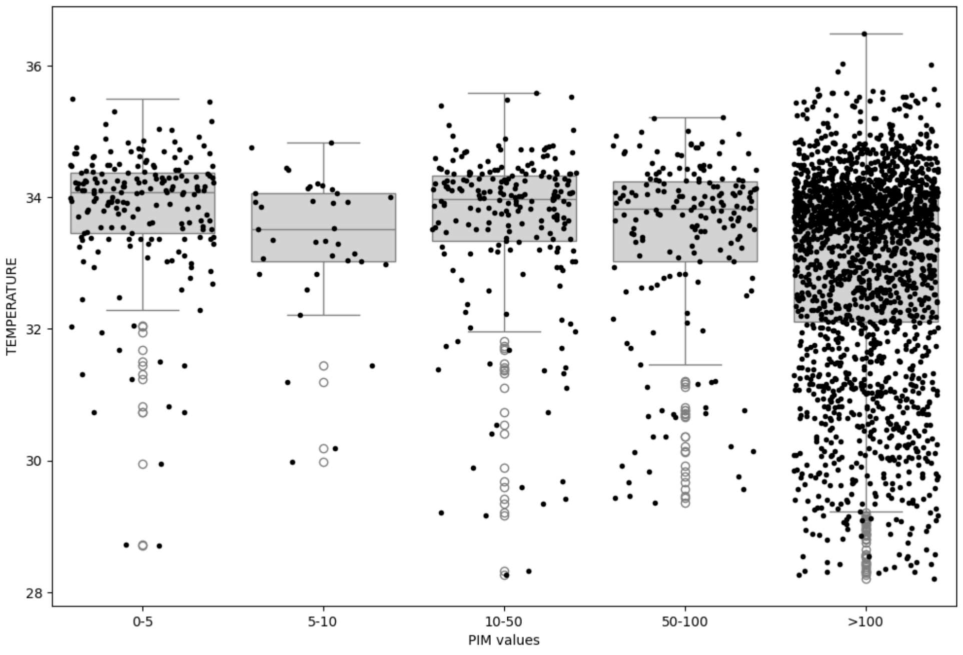


Figure 3.4: Boxplots of Temperature x PIM - patient 04.

Our study involved 244,507 actigraphy recordings, all digitized at 1 sample per second. Of these, 12,941 were classified as OW moments by the Random Forest Classifier (Pilz et al. (2022)), which will be explained in the following subsection, resulting in 231,566 recordings classified as ON moments. A specialist individually prescribed each patient's duration of device use, leading to variations in the total number of days recorded for each patient. This RFC uses entropy to build 30 trees and incorporates automatic feature selection via the out-of-bag score. It mandates a minimum of 2 samples for leaf nodes and 5 for splits. This model identifies extended OW periods and excludes days with over four hours of such intervals, resulting in a dataset comprising 5.29% of OW data.

The methodology for reconstructing actigraphy data can be described in the steps presented in Figure 3.5, where each step is described as:

- **Random Forest Classifier:** A identification of OW moments using the Random Forest algorithm.
- **Cleaning Data, Feature transform and scaling:** Transforming and scaling the features in the data to prepare it for the regression model.
- **Removing OW Remaining Rows*:** Specific data cleaning step to remove OW data on the remaining training data. This step has to be aligned with the filtering data strategies since, in some of those, the presence of the off-wrist moments in some training data can represent the shift to another filtering strategy (as in the case of the Around Gap with DTW method).
- **Filtering Data Strategies:** Application of strategies to filter and refine the data.
- **Regression Strategies:** Application of regression techniques to model the data.

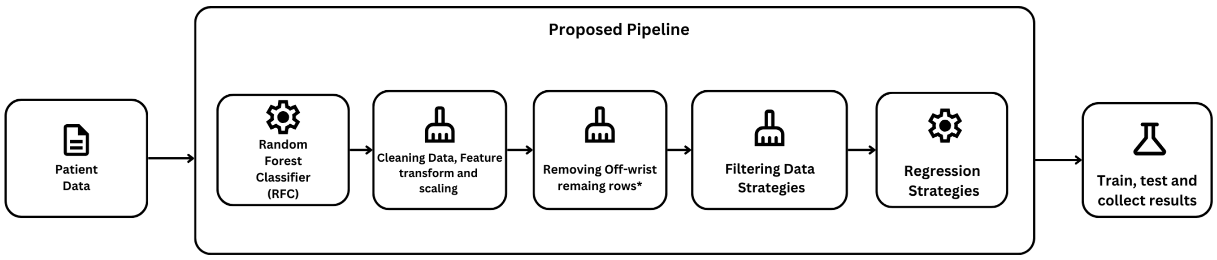

Proposed Pipeline
Patient Data
Random Forest Classifier (RFC)
Cleaning Data, Feature transform and scaling
Removing Off-wrist remaing rows*
Filtering Data Strategies
Regression Strategies
Train, test and collect results

Figure 3.5: Flowchart showing the methodology steps.

3.1 Random Forest Classifier (RFC)

The first problem we identify by analyzing our dataset is that we don't have any label marking our data as moments of OW, but only significant moments marked as 0 for column PIM. As 0 values for column PIM can also represent that the patient does not realize any movement, we can not classify every 0 register for column PIM as OW moments.

In this context, we decided to utilize the model developed by Pilz et al. (2022), which consists of an RFC capable of classifying OW moments. This model is compatible with our dataset, which comes from Condor, the ActTrust device, and adds a new column to our data set, called $ML\_OffWrist\_Prediction$, which contains values 0 and 1, where 1 represents moments classified as OW moments and 0 as ON moments.

3.2 Cleaning Data, Feature transform and scaling

First, to clean the dataset, aiming not to process data that does not have an impact on the model's predictions, we analyzed some patterns and correlations between the variables (presented in the previous section, with the correlation matrix shown in Figure 3.1). First, some variables such as MS, $EVENT$, and $STATE$ presented only 0 values for all patients (which is a shame since a variable as state could indicate if the patient has OW moments on it or not, according to the ActTrust manual and, since its always 0, we have to perform the RFC on the data), so we only removed these columns. Furthermore, columns like $TATn$, $ZCMn$, and $PIMn$ represented the normalized value of the columns TAT, ZCM, and PIM, respectively, so they were also removed.

The $ML\_OffWrist\_Prediction$ was used only to identify the OW moments and, after the filtering strategies, is also removed from the training and test data (since the training data will have only values that were not marked as OW and test data will be marked as ON moments, and the moments marked as OW are the ones who will be imputed). Moreover, the column 'DATE/TIME' was decoded in two columns, $hour$, and day, to utilize the 'DATE/TIME' object information in the training process.

Finally, to perform the normalization process and ensure that each feature contributes equally to the distance calculations, we utilized the MinMaxScaler. This tool efficiently scales the data to a specified range, typically between 0 and 1, which is crucial for maintaining consistency across different feature scales and enhancing the performance of machine learning algorithms that rely on distance metrics.

3.3 Removing Off-wrist remaining rows

This step is responsible for removing the remaining rows marked as OW depending on the filtering data strategies used along with the model. It's important to emphasize this statement since some strategies rely on the presence of OW moments in close registers to the current OW moment being analyzed. Still, other methods don't rely on this, so all the remaining OW rows can be removed in this step.

3.4 Filtering Data Strategies

3.4.1 Vigil periods and M10 approaches

In this approach, we integrated the RFC into our pipeline, assigning it the task of identifying OW moments. In this approach, our primary objective was not to utilize the entire dataset, excluding the OW periods, but to focus on the data from each day with the highest activity levels for the patient. As previously mentioned, Vigil periods typically represent the times of highest activity for the patient. However, because fixed intervals define these periods, there is a risk of missing data if the patient does not exhibit high activity levels within these intervals. This limitation is mitigated by the M10 method, which dynamically defines the interval for each patient each day, thereby accommodating individual activity patterns.

Both approaches have their respective advantages and disadvantages. Suppose the specialist prefers the imputation method to focus on scenarios with a fixed vigil period, thereby maintaining the influence of low or high activity levels on other OW moments of the day. In that case, using the fixed vigil period is likely the most appropriate approach. Conversely, suppose the specialist aims for the imputation process to consistently consider the patient's highest activity levels, resulting in higher activity levels reflected in the imputation process. In that case, the M10 metric as a clipping strategy is likely the best fit. However, it is essential to note that both scenarios will introduce a bias into the imputation process, similar to all previously used methods.

Code 1 shows the structure developed to filter the M10 data:

```
Algorithm 1: Get 10 hours of most excitement for each day
  Input: DataFrame $df$, activity column $activity\_column$
  Output: DataFrame with 10 hours of most excitement for each day
  $epoch \leftarrow \frac{df["DATE/TIME"][1]-df["DATE/TIME"][0]}{60}$;
  $n\_epochs \leftarrow \frac{10\times 60}{epoch}$;
  $days \leftarrow$ unique days from $df["DATE/TIME"]$;
  $activity\_data \leftarrow$ empty DataFrame;
  foreach $day$ in $days$ do
      $data\_today \leftarrow$ filter $df$ by $day$;
      $10\_HOURS\_IN\_ROWS$ = 600 number the minutes that represents 10 hours if
        $data\_today$ has less than $10\_HOURS\_IN\_ROWS$ rows then
          ''' In cases that we have days with less then 10 hours, we won't apply any
            filtering to it. '''
          continue;
      $activity\_sum \leftarrow$ rolling sum of $activity\_column$ over $n\_epochs$;
      $end\_index \leftarrow$ index of maximum value in $activity\_sum$;
      $start\_index \leftarrow end\_index - (n\_epochs - 1)$;
      $activity\_data\_today \leftarrow$ select rows from $start\_index$ to $end\_index$;
      concatenate $activity\_data\_today$ to $activity\_data$;
  return $activity\_data$;
```

3.4.2 Around Gap with DTW approaches

Our current strategy combines the Around Gap and DTW approaches. The choice between these methods depends on whether the patient's data exhibits continuous OW or spaced gap moments. The around-gap strategy is employed when the patient's data contains a constant gap, ensuring data availability with the same length as the gap before and after the OW period.

For instance, if a patient experiences an OW period of 1 hour between 4:00 PM and 6:00 PM, we can collect data for 1 hour before the gap (from 3:00 PM to 4:00 PM) and 1 hour after the gap (from 6:00 PM to 7:00 PM). Thus, our training data will comprise 2 hours (120 minutes), resulting in 120 records, representing a one-minute interval.

In cases where data contains gaps during OW moments, it is not always possible to ensure the availability of sufficient ON data before and after the gap. For instance, if there is an OW period of 40 minutes, followed by 20 minutes of ON data, and then another 20 minutes of OW time, the around-gap strategy cannot be employed because the after-gap data includes OW moments.

In such scenarios, we utilize the DTW approach, which involves identifying similar periods when the OW event occurred but on different days from other weeks. For example, if there is a 30-minute gap between 3:30 PM and 4:00 PM on a Thursday in the first week of the patient's data, we search for the same period on Thursdays of different weeks. If there are more than two occurrences of Thursdays in the dataset, we compute the DTW between these days to identify the most common day. The data from this selected period, corresponding to the OW time, is then used as training data. Code 2 and 3 shows, respectively, the filtering strategy for around gap and DTW:

Algorithm 2: Filter DataFrame by expanding around index gaps

```
Input: DataFrame $df$, DataFrame $gaps\_df$, Integer $multiply\_factor$
Output: Filtered DataFrame and $gaps\_df$
$indexes \leftarrow$ list of indexes from $gaps\_df$;
$range\_previous \leftarrow$ registers of same length before the gap;
$range\_next \leftarrow$ registers of same length after the gap;
$index\_ranges \leftarrow [range\_previous, range\_next]$;
$dfs \leftarrow$ empty list;
foreach $(start, stop)$ in $index\_ranges$ do
    $sliced\_df \leftarrow$ slice $df$ from $start$ to $stop$;
    append $sliced\_df$ to $dfs$;
return $concat(dfs)$, $gaps\_df$;
```

Algorithm 3: Apply Dynamic Time Warping (DTW) filtering strategy

```
Input: DataFrame $df$, DataFrame with gaps $gaps\_df$, Integer
       $MINIMUM\_DAYS = 2$
Output: Tuple containing filtered DataFrames and gaps DataFrames
$gaps\_df \leftarrow$ split $df$ into gaps;
Add column 'day_of_week' to $df$, calculated from $df[''DATE/TIME'']$;
$final\_gaps\_df \leftarrow$ empty list;
$final\_filtered\_dfs \leftarrow$ empty list;
foreach $gap\_df$ in $gaps\_df$ do
    $day\_of\_week \leftarrow$ first element of $gap\_df[''DATE/TIME''].dayofweek$;
    $matching\_days \leftarrow$ list of days in $df$ where $day\_of\_week$ matches and the date is
      different from $gap\_df$'s date;
    if $matching\_days$ is empty then
        continue;
    if $len(matching\_days) \leq MINIMUM\_DAYS$ then
        $final\_day \leftarrow$ first $matching\_day$ where 'ML_OffWrist_Prediction' does not
          contain 1;
    else
        $most\_similar\_day \leftarrow$ day from $matching\_days$ with minimum similarity
          score to $gap\_df$ using DTW;
        $final\_day \leftarrow$ copy of $most\_similar\_day$;
    if $final\_day$ is None then
        continue;
    $start\_time, end\_time \leftarrow$ first and last times in $gap\_df[''DATE/TIME'']$;
    Expand $start\_time$ and $end\_time$ by the length of $gap\_df$;
    $final\_day\_filtered \leftarrow$ filter $final\_day$ by $start\_time$ and $end\_time$;
    if 1 is not in $final\_day\_filtered[''ML\_OffWrist\_Prediction'']$ then
        Append $gap\_df$ to $final\_gaps\_df$;
        Append $final\_day\_filtered$ to $final\_filtered\_dfs$;
Drop 'day_of_week' from $df$;
if $final\_filtered\_dfs$ is empty then
    Raise ValueError: 'No similar days found';
return $final\_filtered\_dfs, final\_gaps\_df$;
```

3.5 Regression Strategies

3.5.1 Autoencoder and TMAE

Our pipeline was developed using the autoencoder framework outlined by Fiorillo et al. (2023). Our methodology utilized the autoencoder for data imputation by feeding the entire dataset as input. Initially, we did not have access to the RFC, which we later used in other pipelines focused on vigil and M10 periods. One major advantage of the autoencoder over other methods is its ability to impute the PIM column—the primary focus of our work—and all other columns in the dataset. This capability is due to the autoencoder having the same number of neurons in its input and output layers.

Code 4 shows the building process of the autoencoder for each patient using the hyperparameter optimization technique:

Algorithm 4: Build Autoencoder Model

```
Input: Hyperparameters $hp$, DataFrame $df$ with features
Output: Compiled Autoencoder model
$input\_dim$ ← number of columns in $df$;
$encoding\_dim$ ← integer value from $hp$ in the range [8, 64] with step sizeeight 8;
$dropout\_rate$ ← float value from $hp$ in the range [0.1, 0.5] with step size 0.1;
$inputs$ ← input layer with shape ($input\_dim$);
$encoded$ ← dense layer with $encoding\_dim$ units and ReLU activation applied to
  $inputs$;
$encoded$ ← dropout layer with rate $dropout\_rate$ applied to $encoded$;
$decoded$ ← dense layer with $input\_dim$ units and sigmoid activation applied to
  $encoded$;
Create autoencoder model with $inputs$ as input and $decoded$ as output;
Compile autoencoder with Adam optimizer, mean squared error (MSE) loss, and the
  following metrics: MAE, RMSE, $R^2$, MASE.
return autoencoder model;
```

The TMAE followed a similar structure, just changing the layers so it has the masking operation, the positional encoder, and the transformer encoder and decoder. We masked the data using values in a range of 15% to 40%, along with the hyperparameter tunning techniques. Code 5 shows the training algorithm for the TMAE.

Algorithm 5: Train TMAE with Early Stopping and Validation

```
Input: $model$: Model, $train\_loader$: DataLoader, $val\_loader$: DataLoader,
       $num\_epochs$: Integer, $learning\_rate$: Float, $patience$: Integer (default 10)
Output: $model$: Model, $best\_val\_loss$: Float
Initialize loss function $criterion \leftarrow$ Mean Squared Error Loss;
Initialize optimizer $optimizer \leftarrow$ Adam optimizer with $learning\_rate$;
Set $device \leftarrow$ cuda if available else cpu; Move $model$ to $device$;
$best\_val\_loss \leftarrow \infty$; $patience\_counter \leftarrow 0$; $best\_model\_state \leftarrow$ None;
for $epoch$ from $1$ to $num\_epochs$ do
    Set $model$ to training mode; foreach $batch$ in $train\_loader$ do
        $batch \leftarrow$ detach and move $batch[0]$ to $device$;
        $masked\_batch, \_ \leftarrow$ apply masking to $batch$ with mask ratio;
        $reconstructed \leftarrow model(masked\_batch)$;
        $loss \leftarrow criterion(reconstructed, batch)$; Zero gradients in $optimizer$;
        Backpropagate $loss$; Update $optimizer$;
    Set $model$ to evaluation mode; $val\_loss \leftarrow 0$; Initialize empty lists $y\_true\_list$,
     $y\_pred\_list$; foreach $val\_batch$ in $val\_loader$ do
        $val\_batch \leftarrow$ detach and move $val\_batch[0]$ to $device$; $masked\_val\_batch, \_ \leftarrow$
         apply masking to $val\_batch$ with mask ratio;
        $reconstructed \leftarrow model(masked\_val\_batch)$;
        $val\_loss \leftarrow val\_loss + criterion(reconstructed, val\_batch)$;
        Append $val\_batch$ to $y\_true\_list$;
        Append $reconstructed$ to $y\_pred\_list$;
    $val\_loss \leftarrow val\_loss/$ length of $val\_loader$;
    $y\_true\_concat \leftarrow$ concatenate $y\_true\_list$;
    $y\_pred\_concat \leftarrow$ concatenate $y\_pred\_list$;
    Compute metrics $mse$, $mae$, $rmse$, $r2$, $mase$ using $y\_true\_concat$ and
     $y\_pred\_concat$;
    Print epoch details including $loss$, $val_loss$, and metrics;
    if $val\_loss < best\_val\_loss$ then
        $best\_val\_loss \leftarrow val\_loss$; $patience\_counter \leftarrow 0$; $best\_model\_state \leftarrow$
         current state of $model$;
    else
        $patience\_counter \leftarrow patience\_counter + 1$;
        if $patience\_counter \geq patience$ then
            Print "Early stopping at epoch $epoch$";
            break;
if $best\_model\_state$ is not None then
    Load $best\_model\_state$ into $model$;
    Print "Loaded the best model based on validation loss.";
return $model, best\_val\_loss$;
```

3.5.2 Machine Learning with Hyperparameter Tunning

Firstly, it is essential to note that this step was utilized with the filtering methods: vigil periods, M10, and the around Gap with DTW (Dynamic Time Warping). The ML approach employs all the regression models, excluding the autoencoder mentioned in section 2, to function collectively as a grouped model. This grouped model leverages the strengths of multiple regression techniques to improve the accuracy and robustness of the predictions. Each model within the group is trained using the patient's data. The most effective model, determined by statistical metrics, also informed in section 2, is chosen as the final predictor for the data imputation process. Also, the DummyRegressor was trained for each patient only as a baseline method, so we can compare the analyzed models and the basic techniques that the DummyRegressor brings.

Each model is subjected to the GridSearch and RandomizedSearch hyperparameter tuning algorithms during this phase. These algorithms train the data using various combinations of hyperparameters for each model, adhering to the optimization strategies of each respective algorithm. By fine-tuning the models through these hyperparameter optimization techniques, the performance of each model is maximized, leading to more accurate and reliable predictions. This meticulous tuning process is crucial for enhancing the model's ability to generalize from the training data to unseen data, thereby improving the overall effectiveness of the ML strategy in the data imputation process. The hyperparameters for each model are presented in Table 3.1.

Table 3.1: Models Hyperparameters

Models	Hyperparameters
RFR	bootstrap, n_estimators, max_features
SVR	C, kernel, gamma
XGBRegressor	n_estimators, learning_rate, max_depth, objective, min_child_weight
	subsample, colsample_bytree
Decision Tree	max_depth, min_samples_split
KNN Regressor	n_neighbors, weights
Linear Regressor	fit_intercept

Code 6 shows the training process using KFold and training with both GridSearch and RandomizedSearch for all the presented models in Section 2.

Algorithm 6: Validate Models Using GridSearch and RandomizedSearch

```
Input: Training data $X\_train$, $y\_train$; Test data $X\_test$, $y\_test$; List of models with
       parameter grids
Output: Best model, best score, second best model, second best score, worst model,
        worst score
$kfold \leftarrow$ K-fold cross-validation with five splits and shuffle enabled;
foreach model in models do
    $model\_name \leftarrow$ name of the current model;
    $model \leftarrow$ current model;
    $param\_grid \leftarrow$ parameter grid for GridSearch;
    $param\_dis \leftarrow$ parameter distribution for RandomizedSearch;
    Perform GridSearchCV on $model$ using $param\_grid$ with scoring
      'neg_mean_squared_error' and 5-fold CV;
    Fit the GridSearch model on training data;
    Predict values on test data using GridSearch model;
    Compute performance metrics: MAE, RMSE, R², MASE for GridSearch
      predictions;
    if $grid\_rmse > best\_score$ then
        Update $second\_best\_model$ and $second\_best\_score$ with previous best values;
        Set $best\_model$ and $best\_score$ to GridSearch best estimator and $grid\_rmse$;
    else if $grid\_rmse > second\_best\_score$ then
        Update $second\_best\_model$ and $second\_best\_score$ with GridSearch best
          estimator and $grid\_rmse$;
    else if $grid\_rmse < worst\_score$ then
        Update $worst\_model$ and $worst\_score$ with GridSearch best estimator and
          $grid\_rmse$;
    Perform RandomizedSearchCV on $model$ using $param\_dis$ with scoring
      'neg_mean_squared_error' and 5-fold CV;
    Fit the RandomizedSearch model on training data;
    Predict values on test data using RandomizedSearch model;
    Compute performance metrics: MAE, RMSE, R², MASE for RandomizedSearch
      predictions;
    if $randomized\_rmse > best\_score$ then
        Update $second\_best\_model$ and $second\_best\_score$ with previous best values;
        Set $best\_model$ and $best\_score$ to RandomizedSearch best estimator and
          $randomized\_rmse$;
    else if $randomized\_rmse > second\_best\_score$ then
        Update $second\_best\_model$ and $second\_best\_score$ with RandomizedSearch
          best estimator and $randomized\_rmse$;
    if $randomized\_rmse < worst\_score$ then
        Update $worst\_model$ and $worst\_score$ with RandomizedSearch best estimator
          and $randomized\_rmse$;
return best_model, best_score, second_best_model, second_best_score,
  worst_model, worst_score;
```

3.5.3 Final Considerations

In this chapter, we presented a methodology for reconstructing actigraphy data affected by Off-Wrist (OW) periods. The approach began with Exploratory Data Analysis (EDA) to understand the dataset's structure and correlations, leading to the removal of irrelevant features like MS, $EVENT$, $STATE$, and normalized variables. We employed an RFC to identify OW moments accurately, adding an $ML\_OffWrist\_Prediction$ column to distinguish between OW and On-Wrist (ON) periods.

Data cleaning involved decoding the 'DATE/TIME' column into 'hour' and 'day' components and normalizing features using MinMaxScaler. We implemented filtering strategies tailored to our objectives: Vigil periods and M10 approaches focused on high-activity intervals, while the Around Gap with Dynamic Time Warping (DTW) method filled gaps by finding similar activity patterns on different days.

For data imputation, we utilized an Autoencoder to reconstruct missing values across all features and applied machine learning models—including Random Forest Regressor, Support Vector Regressor, XGBoost, Decision Trees, and K-Nearest Neighbors—optimized through hyperparameter tuning. This combination of strategies aimed to enhance the accuracy and reliability of actigraphy analyses by effectively handling missing data due to OW periods.

Our methodology offers a robust framework for improving patient activity monitoring, providing scalable solutions applicable to various research settings within sleep and circadian rhythm studies.

Chapter 4

Results

All the results obtained for autoencoder were collected using the platform Google Colab Pro (payed version of Google Colab), which had 12.67 GB of RAM, 107.7 GB of hard disk, and his default GPU, which is an NVIDIA RTX Tesla K80 with 12GB of VRAM. The results using the Essemble methods for M10 and Vigil times were using a laptop with an 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz CPU, 16 GB DDR4 of RAM, and 768 GB of SSD. The M10 and DTW results were collected using a notebook with 64 GB of DDR5 and a processor from Intel, the 13th Gen Intel(R) Core(TM) i7-13700HX.

Following the RFC step to determine OW duration, we included only 9 out of the 23 available patients in the study. We excluded fourteen patients because they either lacked OW episodes or had more than four hours of OW time daily. Figure 4.1 illustrates the PIM (Patient Impact Measure) values for Patient 7 (P7) across nine days. The RFC recorded 1,579 OW time, totaling approximately 26 hours and 32 minutes, shown in Figure 4.2. The frequency of these OW incidents could significantly affect a specialist's evaluation, leading to the exclusion of the days with more than 4 hours of OW for this patient. After this step, there were still some days with less than 4 hours of OW moments, which made the patient still eligible for the analysis.

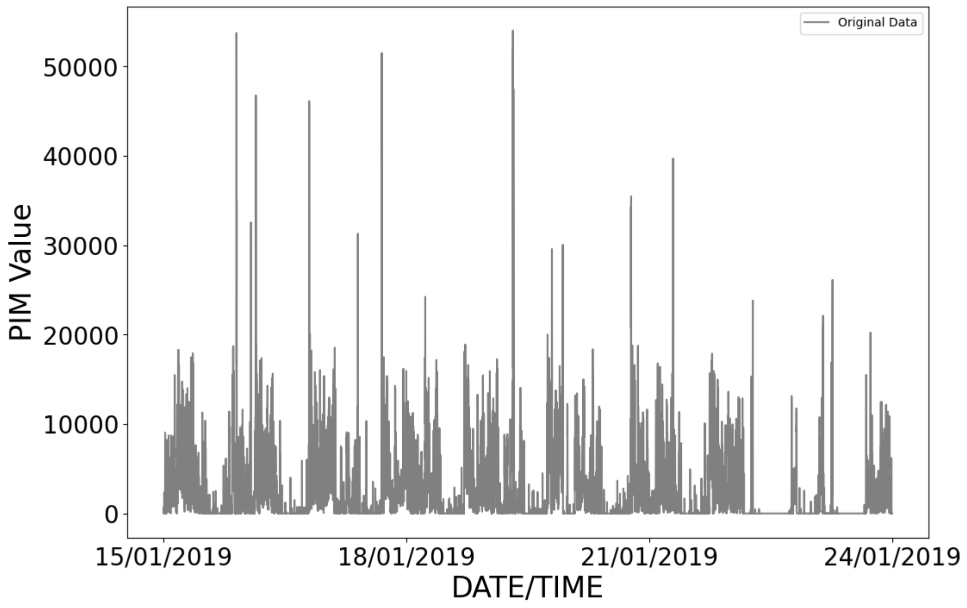


Figure 4.1: PIM original data value from Patient 7.

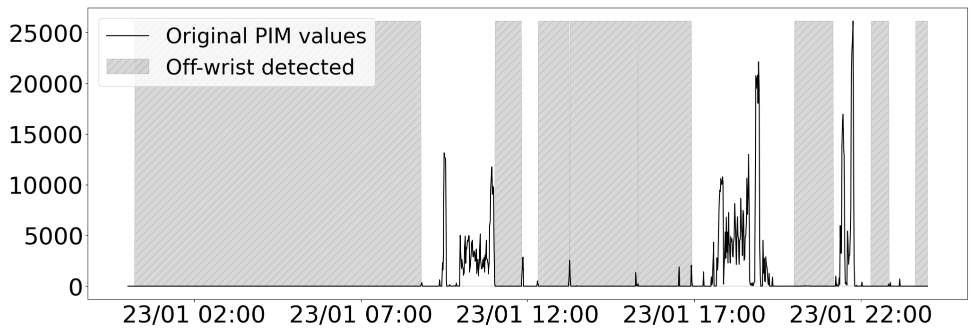


Figure 4.2: OW detected moments for Patient 7 using the RFC.

In the following subsections, we will present three tables showing, respectively, the statistical metrics for each imputed patient and the Wilcoxon tests (this type of test was chosen since the PIM data does not follow a Gaussian distribution; we tested all the patients with the Shapiro-Wilk test to identify if they followed a Gaussian distribution), and the Actigraphy metrics provided by the Non-Parametric Measures of Actigraphy Data (nparACT), using the language R.

The tables where we present the Wilcoxon results will show both T-Statistic and P-value results. The T-Statistic is used to assess the evidence against the null hypothesis. In the case of the Wilcoxon signed-rank test, the null hypothesis typically states that the median difference between the paired samples is zero. A smaller stat value indicates more substantial evidence against the null hypothesis, suggesting a significant difference between the paired samples.

The p-value is used to determine the statistical significance of the test results. A common threshold for significance is 0.05:

- If P-Value < 0.05: There is sufficient evidence to reject the null hypothesis, suggesting a statistically significant difference between the paired samples.

- If P-Value $>=$ 0.05: There is not enough evidence to reject the null hypothesis, suggesting that any observed differences could reasonably occur by chance under the null hypothesis.

4.1 Autencoder with Full Data

The Autoencoder approach shows good performance metrics across most patients, as shown in Table 4.2. The mean squared error (MSE) and mean absolute error (MAE) values are low, with the R² values nearing 1.00, indicating a highly accurate model. Using complete data with the Autoencoder allows for precise reconstruction, as evidenced by metrics such as RMSE and MASE. The NaN value in MASE for patient 21 was caused by a division by zero since the metric for calculating the MASE is a custom method we developed. Since the denominator represents the Mean Absolute Naive Error (MANE), calculated as the average of the absolute differences between consecutive actual observations, normalizing the data could result in all values becoming the same. In this case, the differences between successive observations would be zero, creating a constant time series. Also, we have a different number of neurons and weights for each patient for each layer, but a prevalent pattern of the following layers, with a dense dropout and another dense layer, is shown in Table 4.1.

Table 4.1: Autoencoder Models Structure.

Patient	Input Neurons	Dense Neurons	Dropout Rate	Output Neurons
04	12	64	10%	12
05	12	40	10%	12
07	12	48	10%	12
15	12	56	10%	12
16	12	40	10%	12
21	12	56	10%	12
22	12	48	10%	12
23	12	48	20%	12
25	12	64	10%	12

The Wilcoxon test results, presented in Table 4.3, support the statistical significance of the Autoencoder's performance, with p-values indicating strong evidence against the null hypothesis across most patients. For instance, patient 07 has a solid result with a T-statistic of 0.0 and a p-value of 1.3231×10^{-259}, indicating a highly significant difference.

Actigraphy metrics using nparACT in Table 4.4 reveal that the reconstructed data (indicated by sub-index *R*) closely mirrors the original data (*O*), with minor variations in Interdaily Stability (IS) and Intradaily Variability (IV), which reflect the Autoencoder's ability to capture temporal patterns in activity data effectively.

Figure 4.3 shows the imputed values alongside the ON values from patient 4. For clarity, we will mainly focus on presenting the results for patient 4 in this and the following subsections.

Table 4.2: Autoencoder Performance Metrics for Patients. This method was developed using and trained using cross-validation, with five folds and hyperparameter optimization techniques, using Keras-tuner as a library.

Patient	MSE	MAE	RMSE	R²	MASE
04	4.665×10^{-5}	0.00345	0.00683	0.99907	0.00573
05	8.174×10^{-5}	0.00455	0.00904	0.99862	0.00595
07	6.109×10^{-5}	0.00371	0.00782	0.99860	0.00474
15	1.210×10^{-4}	0.00581	0.01100	0.99704	0.00684
16	8.450×10^{-5}	0.00455	0.00919	0.99862	0.00852
21	6.158×10^{-5}	0.00428	0.00785	0.99891	NaN
22	9.294×10^{-5}	0.00493	0.00964	0.99827	0.00706
23	3.768×10^{-4}	0.01196	0.01941	0.99522	0.02108
25	4.457×10^{-5}	0.00307	0.00668	0.99898	0.00457

Table 4.3: Wilcoxon Test for Patients Using Autoencoder

Patient	T-Statistic	P-Value
04	0.0	7.5778×10^{-12}
05	0.0	1.7322×10^{-6}
07	0.0	1.3231×10^{-259}
15	0.0	6.6916×10^{-118}
16	0.0	3.5595×10^{-13}
21	557.0	1.2640×10^{-45}
22	0.0	3.1480×10^{-179}
23	0.0	1.6410×10^{-12}
25	0.0	5.8354×10^{-19}

The autoencoder method maintained the proportion between the maximum and minimum values of the region without making the values too low or exceeding the maximum PIM value for that patient in that region.

Table 4.4: nparACT Results for Autoencoder: the sub-indices *O* means Original Data and sub-indices *R* means Reconstructed Data. White and gray columns better visualize the same metrics in original and imputed datasets.

Patient	**IS**$_O$	**IS**$_R$	**IV**$_O$	**IV**$_R$	**RA**$_O$	**RA**$_R$	**L5**$_O$**(mg)**	**L5**$_R$**(mg)**	**M10**$_O$**(mg)**	**M10**$_R$**(mg)**
04	0.3	0.28	0.47	1.21	0.92	0.81	158.48	348.66	3689.82	3304.07
05	0.28	0.49	1.21	0.76	0.81	0.9	348.66	227.43	3304.07	4409.29
07	0.49	0.44	0.76	0.93	0.9	0.95	227.43	89.96	4409.29	3583.15
15	0.51	0.53	0.91	0.77	0.92	0.97	143.97	80.46	3306.32	5166.9
16	0.53	0.6	0.78	1.03	0.97	0.87	80.46	206.4	5166.9	2863.02
21	0.6	0.37	1.03	1.12	0.87	0.87	206.4	302.7	2855.47	4522.12
22	0.4	0.4	1.36	1.36	0.92	0.92	148.5	150.53	3422.83	3435.47
23	0.31	0.5	0.73	0.63	0.83	0.89	339.42	146.94	3749.18	2487.73
25	0.47	0.47	0.83	0.82	0.85	0.85	210.04	210.04	2562.18	2565.95

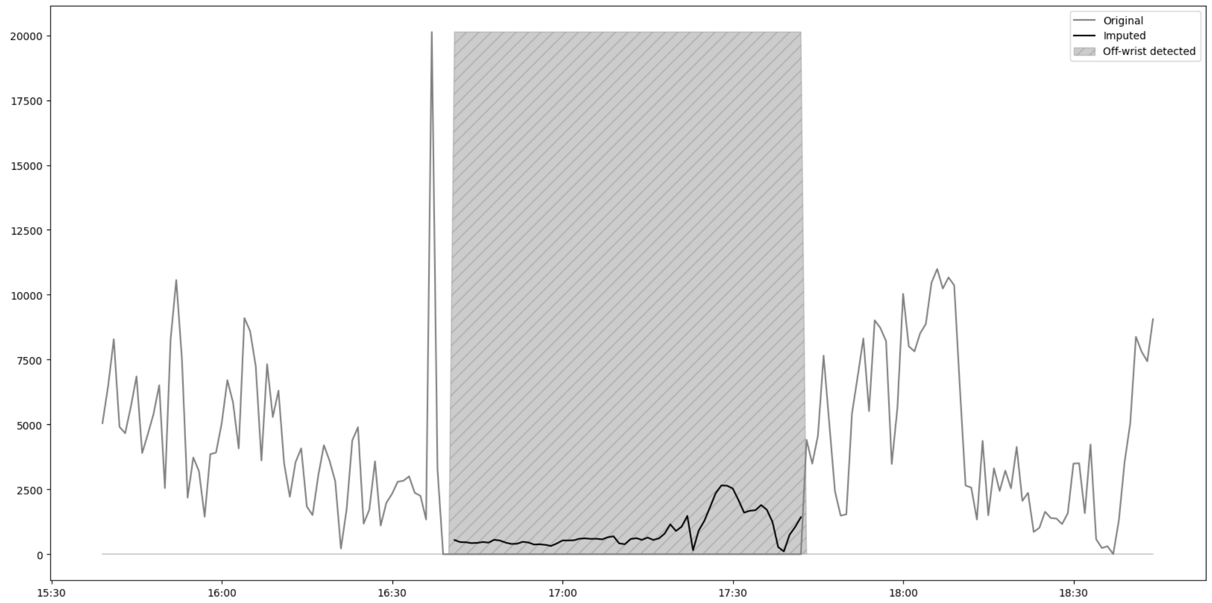


Figure 4.3: Results for patient 4 using Autoencoder method.

4.2 TMAE with Full Data

The TMAE approach demonstrates consistent performance metrics across a range of patients, as shown in Table 4.5. A low MSE, MAE, and RMSE values across most patients should indicate precise predictions. However, the R^2 for all predictions is low or low, as in the case of patient 23, which means a not-so-good fit of the model with the data. The MASE provides additional insight into accuracy relative to naive predictions, with values above one indicating that the model performs poorly. Different from the Autoencoder, the TMAE followed different numbers of layers for transformer layers precisely but started with a dense dropout and finished with another dense layer for all patients.

The Wilcoxon test presented in Table 4.7 shows statistically significant results across all

Table 4.5: TMAE Performance Metrics for Patients. This method was developed using and trained using cross-validation, with also five folds and hyperparameter optimization techniques, using Optuna as a library.

Patient	MSE	MAE	RMSE	R²	MASE
04	0.0028	0.0183	0.0532	0.5694	1.4925
05	0.0018	0.0175	0.0429	0.6789	1.1980
07	0.0013	0.0159	0.0364	0.7012	1.3683
15	0.0016	0.0184	0.0403	0.6959	1.1485
16	0.0009	0.0138	0.0305	0.6943	1.1678
21	0.0017	0.0174	0.0407	0.2649	1.2025
22	0.0026	0.0201	0.0509	0.7022	1.4857
23	0.0053	0.0399	0.0731	0.0894	1.5052
25	0.0020	0.0203	0.0444	0.6757	1.2084

Table 4.6: TMAE Models Structure.

Patient	Input Neurons	Dense Neurons	Dropout Rate	Transformer Encoder (Layers, Heads, Units)	Output Neurons	Total Parameters
04	12	32	10%	(2, 2, 32)	12	2,956
05	12	48	10%	(3, 3, 48)	12	5,782
07	12	64	10%	(4, 4, 64)	12	9,612
15	12	40	10%	(2, 2, 40)	12	3,504
16	12	56	10%	(3, 3, 56)	12	6,332
21	12	64	10%	(4, 4, 64)	12	10,192
22	12	48	10%	(3, 3, 48)	12	5,716
23	12	72	50%	(5, 5, 72)	12	15,428
25	12	80	10%	(6, 6, 80)	12	21,328

patients, with varying T-statistics, meaning there are still some significant differences between the original and imputed data.

The actigraphy metrics in Table 4.8 demonstrate notable discrepancies between the original and reconstructed data across most metrics. Significant changes are observed in IS, IV, RA, L5, and M10 values, indicating that the reconstructed data deviates substantially from the original measurements. These variations suggest potential challenges in maintaining data fidelity within the reconstruction process for these metrics.

]

We can see the impact of a low R^2 in Figure 4.4, which shows that the values for this patient remained close to 0, keeping almost the same behavior as we have an OW.

Table 4.7: Wilcoxon Test for Patients Using TMAE

Patient	T-Statistic	P-Value
04	0.0	5.835×10^{-19}
05	0.0	1.734×10^{-6}
07	1134.0	1.137×10^{-258}
15	107880.0	7.939×10^{-4}
16	0.0	3.559×10^{-13}
21	0.0	3.676×10^{-48}
22	44775.0	1.517×10^{-129}
23	1.0	1.718×10^{-12}
25	0.0	5.835×10^{-19}

Table 4.8: nparACT Results for TMAE

Patient	**IS$_O$**	**IS$_R$**	**IV$_O$**	**IV$_R$**	**RA$_O$**	**RA$_R$**	**L5$_O$(mg)**	**L5$_R$(mg)**	**M10$_O$(mg)**	**M10$_R$(mg)**
01	0.28	0.27	1.21	1.2	0.81	0.81	348.66	348.66	3304.07	3261.23
02	0.49	0.49	0.76	0.76	0.9	0.9	227.43	227.43	4409.29	4367.62
03	0.43	0.32	0.92	0.86	0.96	0.97	64.51	42.44	3544.86	3045.06
04	0.53	0.48	0.78	0.72	0.97	0.97	80.46	64.45	5166.9	4927.33
05	0.6	0.59	1.03	1.03	0.87	0.87	206.4	206.4	2855.47	2855.47
06	0.37	0.36	1.12	1.09	0.87	0.87	301.71	301.71	4521.83	4509.21
07	0.4	0.35	1.36	1.26	0.92	0.92	148.5	130.22	3422.83	3154.41
08	0.45	0.44	0.58	0.71	0.89	0.88	146.94	146.94	2487.73	2228.26
09	0.47	0.46	0.83	0.82	0.85	0.85	210.04	210.04	2562.18	2535.96

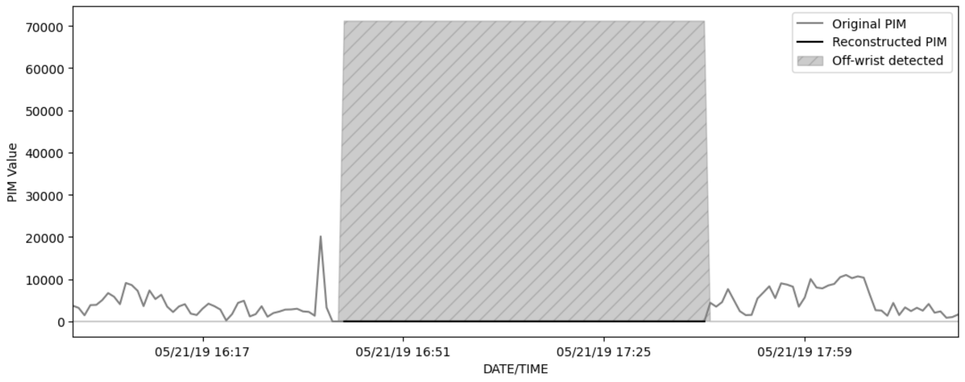


Figure 4.4: Results for patient 4 using TMAE method.

4.3 ML Models with M10

ML Models with the M10 filtering technique (Tables 4.9, 4.10, and 4.11) also demonstrate high performance with minimal errors and high R² values. This method utilizes a combination of models, predominantly XGBRegressor, which is the best model for most patients.

The performance metrics indicate that the ML method handles diverse patient data well, as

Table 4.9: ML Models with M10 Performance Metrics for Patients

Patient	MSE	MAE	RMSE	R²	MASE	Best Model
04	0.00	0.00	0.01	0.98	7645263612.40	DecisionTreeRegressor
05	0.00	0.01	0.01	0.99	4035495354.88	XGBRegressor
07	0.00	0.00	0.01	0.99	24719687124.48	RandomForestRegressor
15	0.00	0.01	0.01	0.98	165095448966.42	XGBRegressor
16	0.00	0.01	0.01	0.99	168646853054.68	XGBRegressor
21	0.00	0.01	0.02	0.99	22310086002.98	XGBRegressor
22	0.00	0.01	0.01	0.98	34967882319.29	XGBRegressor
23	0.00	0.02	0.04	0.94	2783996446055.05	XGBRegressor
25	0.00	0.01	0.01	0.99	68551396104.05	XGBRegressor

seen in the high R², low MAE, and RMSE across the board. Although all patients have a high value of MASE, the forecasting model is performing poorly compared to the naive benchmark for this method.

The Wilcoxon test shows statistically significant results across all patients, with varying T-statistics, reinforcing the model's effectiveness in improving predictive accuracy with the M10 filtering approach. For instance, patient 07 shows a T-statistic of 1608.0 with a p-value of 8.660×10^{-7}, indicating effective enhancement from the ML model.

Table 4.10: Wilcoxon Test for Patients Using ML Models with M10

Patient	T-Statistic	P-Value
04	0.0	3.523×10^{-9}
05	0.0	7.632×10^{-3}
07	1608.0	8.660×10^{-7}
15	104.0	3.811×10^{-10}
16	818.0	1.291×10^{-2}
21	0.0	1.553×10^{-7}
22	0.0	5.137×10^{-10}
23	0.0	1.229×10^{-5}
25	0.0	7.691×10^{-8}

NparACT analysis indicates consistent data reconstruction with the original, with similar values for metrics like IS and IV. The M10-filtered ML models tend to slightly under or overestimate specific metrics, such as L5 and M10, but maintain overall reliability.

Figure 4.5 shows that the values for this patient remained the same, close to 0, with the value of 5.613946 exactly.

Table 4.11: nparACT Results for ML model with M10: the sub-indices *O* means Original Data and sub-indices *R* means Reconstructed Data. White and gray columns better visualize the same metrics in original and imputed datasets.

Patient	**IS**$_O$	**IS**$_R$	**IV**$_O$	**IV**$_R$	**RA**$_O$	**RA**$_R$	**L5**$_O$**(mg)**	**L5**$_R$**(mg)**	**M10**$_O$**(mg)**	**M10**$_R$**(mg)**
04	0.28	0.1	1.21	1.4	0.81	0.3	348.66	1827.74	3304.07	3421.05
05	0.3	0.3	0.47	0.47	0.92	0.92	158.48	158.48	3689.82	3689.82
07	0.49	0.49	0.76	0.76	0.9	0.9	227.43	227.43	4409.29	4409.29
15	0.53	0.32	0.78	0.95	0.97	0.34	80.46	2422.98	5166.9	4868.62
16	0.6	0.6	1.03	1.03	0.87	0.87	206.4	206.4	2855.47	2855.47
21	0.4	0.4	1.36	1.36	0.92	0.92	148.5	148.5	3422.83	3422.83
22	0.31	0.31	0.73	0.73	0.83	0.83	339.42	339.42	3749.18	3749.18
23	0.47	0.47	0.83	0.83	0.85	0.85	210.04	210.04	2562.18	2562.18
25	0.6	0.6	0.71	0.71	0.96	0.96	149.74	149.74	6854.28	6854.28

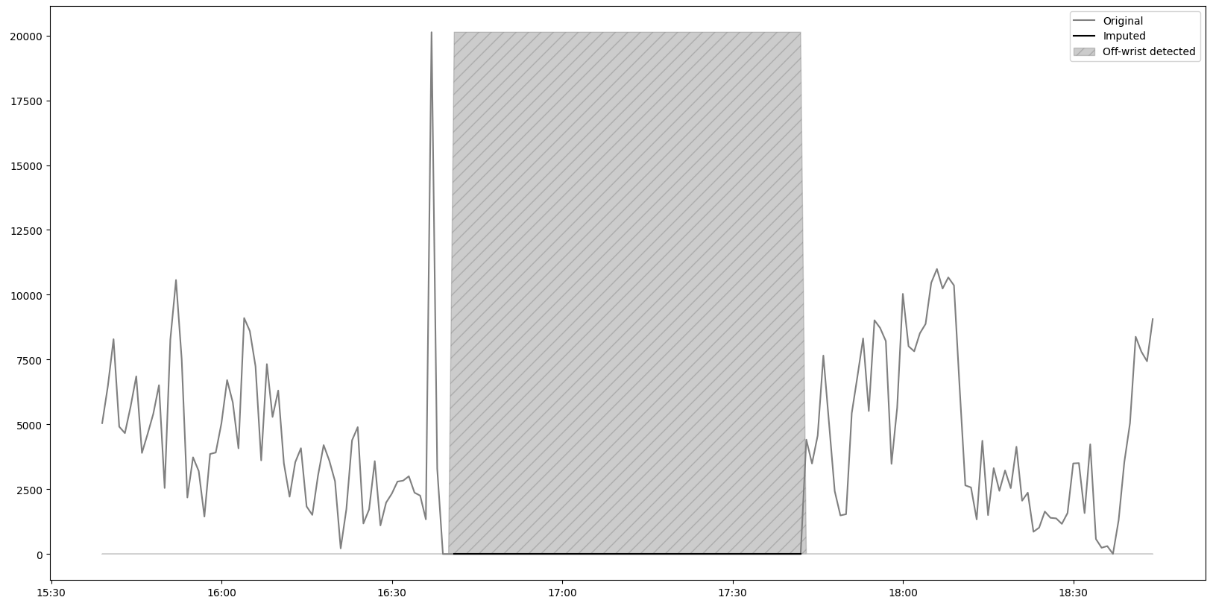


Figure 4.5: Results for patient 4 using M10 method.

4.4 ML Models with Around Gap with DTW

The ML models using the Around Gap with DTW method, detailed in Tables 4.12, 4.13, and 4.14, achieve competitive performance metrics with low MSE, MAE, and high R² values, demonstrating its capacity to predict patient data accurately.

A notable observation is that while the MASE values are generally lower, indicating stable predictive performance, there is a significant outlier in patient 05 with an extraordinarily high MASE. This outlier suggests that the model might be sensitive to specific data characteristics or distributional shifts.

The Wilcoxon test results corroborate the model's performance, with significant p-values across most patients, albeit with some variability in T-statistics. For example, patient 21 exhibits

Table 4.12: Model Performance Metrics for Patients

Patient	MSE	MAE	RMSE	R²	MASE	Best Model
04	5.90	0.02	0.04	0.95	3.58	LinearRegression
05	0.00	0.01	0.02	0.99	1241395892094.03	LinearRegression
07	0.00	0.01	0.02	0.99	0.24	XGBRegressor
15	0.00	0.03	0.04	0.98	480016592680.68	XGBRegressor
16	2.15	0.01	0.01	1.00	1315259802431.74	LinearRegression
21	0.00	0.01	0.02	0.99	0.10	RandomForestRegressor
22	0.00	0.01	0.01	0.96	0.39	XGBRegressor
23	0.00	0.02	0.03	0.98	5890949679407.39	LinearRegression
25	0.00	0.02	0.03	0.98	0.48	DecisionTreeRegressor

a T-statistic of 0.0 with a p-value of 1.553×10^{-7}, confirming the method's effectiveness in aligning patient data with DTW.

Table 4.13: Wilcoxon Test for Patients Using ML Models with DTW

Patient	T-Statistic	P-Value
04	0.0	3.523×10^{-9}
05	0.0	7.632×10^{-3}
07	1608.0	8.660×10^{-7}
15	104.0	3.811×10^{-10}
16	818.0	1.291×10^{-2}
21	0.0	1.553×10^{-7}
22	0.0	5.137×10^{-10}
23	0.0	1.229×10^{-5}
25	0.0	7.691×10^{-8}

NparACT metrics highlight the impact of DTW in aligning activity patterns, with reconstructed data generally matching the original. Variations in metrics like IS and IV are present but do not significantly deviate from the expected values, indicating the proposed technique may not present some differences from the original dataset after performing its steps.

Figure 4.6 illustrates that the values for this patient varied, fitting well within the data context. Low values followed this at the beginning of the imputed values, as they originated from a drop in the values from the previous context and connected with the following context at the end of the series.

Table 4.14: nparACT Results for ML models with DTW and Around Gap: the sub-indices *O* means Original Data and sub-indices *R* means Reconstructed Data. White and gray columns better visualize the same metrics in original and imputed datasets.

Patient	**IS**$_O$	**IS**$_R$	**IV**$_O$	**IV**$_R$	**RA**$_O$	**RA**$_R$	**L5**$_O$**(mg)**	**L5**$_R$**(mg)**	**M10**$_O$**(mg)**	**M10**$_R$**(mg)**
04	0.3	0.48	0.47	0.68	0.92	0.92	158.48	189.59	3689.82	4751.14
05	0.28	0.28	1.21	1.21	0.81	0.81	348.66	348.66	3304.07	3304.07
07	0.49	0.49	0.76	0.76	0.9	0.9	227.43	231.42	4409.29	4409.29
15	0.53	0.57	0.78	0.77	0.97	0.97	80.46	79.11	5166.9	5477.73
16	0.6	0.61	1.03	1.05	0.87	0.87	206.4	206.4	2855.47	2858.54
21	0.4	0.16	1.36	1.43	0.92	1	148.5	240.86	3422.83	196603.89
22	0.31	0.53	0.73	0.76	0.83	0.95	339.42	120.3	3749.18	4733.17
23	0.47	0.47	0.83	0.83	0.85	0.85	210.04	210.04	2562.18	2565.69
25	0.6	0.6	0.71	0.71	0.96	0.96	149.74	149.74	6854.28	6862.58

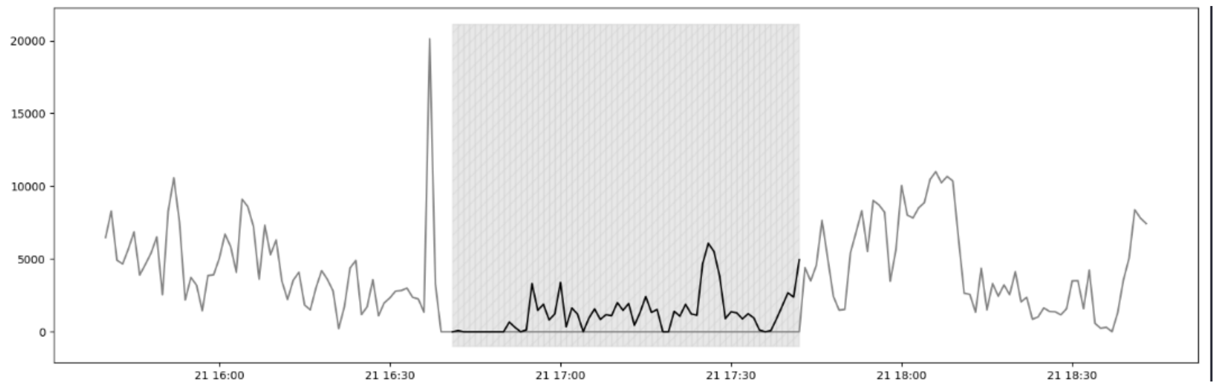


Figure 4.6: Results for patient 4 using Around + DTW method.

4.5 SHAP Evaluations of ML Models

To better understand the impact of other variables in the learning process of the ML Models, we performed SHAP evaluation in the results for each one of the nine eligible patients. In some cases, as for patient 04 and patient 21, the results seemed that every variable had a similar impact in patient four since the values didn't present any deviation for more or less as shown in Figures 4.7 and 4.8 for patient four, and in Figures 4.9 and 4.10 for patient 21.

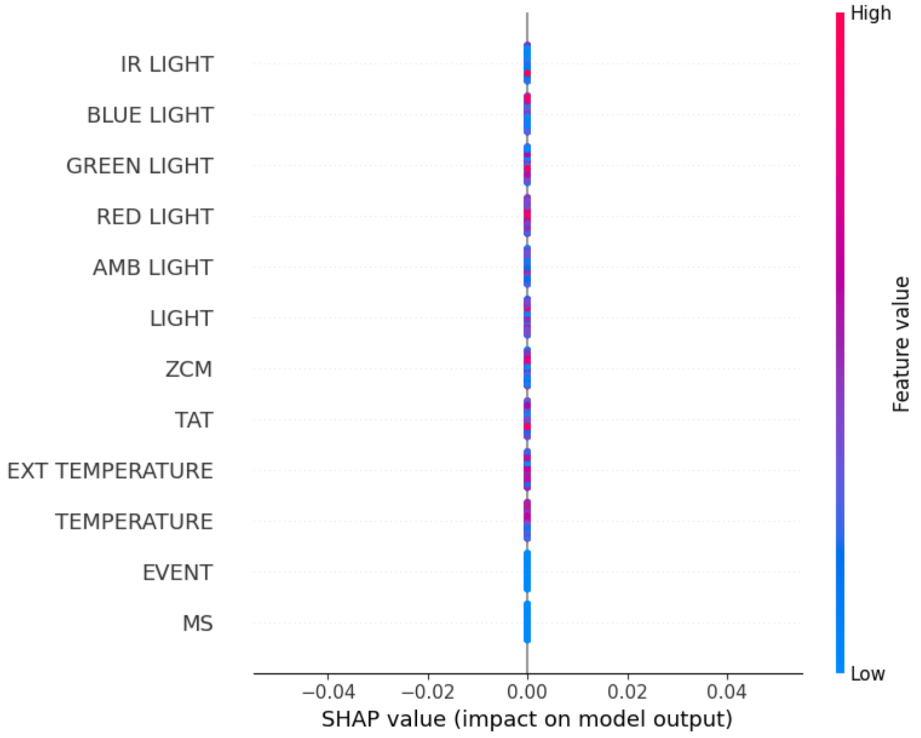


Figure 4.7: Summary Plot for patient 4.

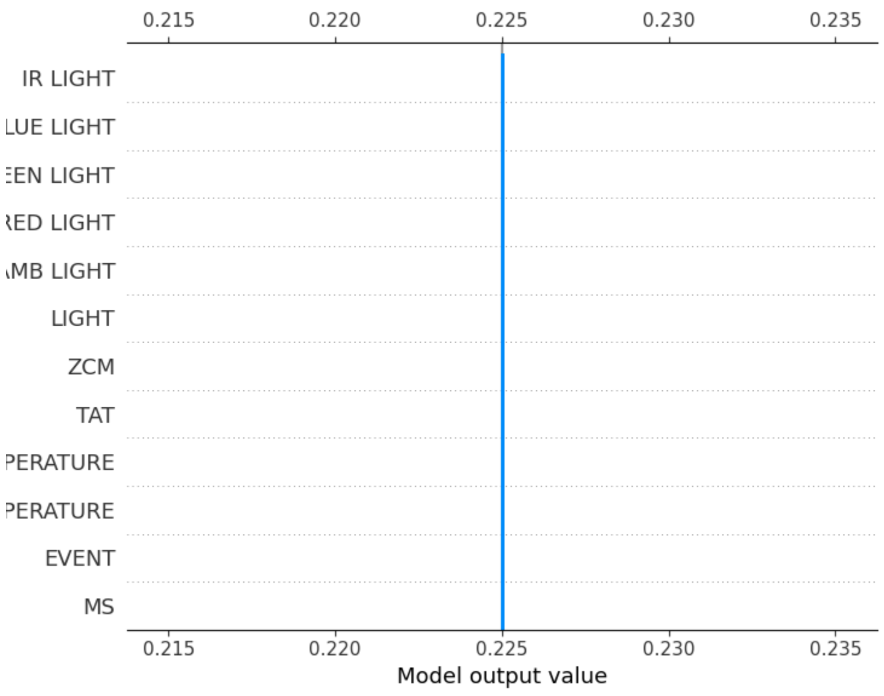


Figure 4.8: Decision Plot for patient 4.

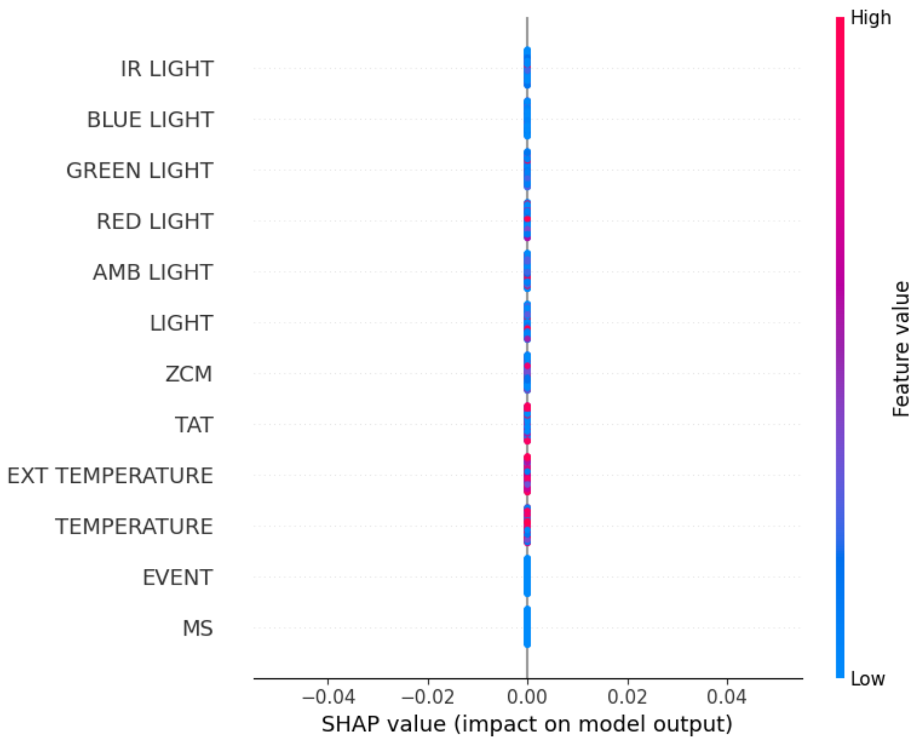


Figure 4.9: Summary Plot for patient 21.

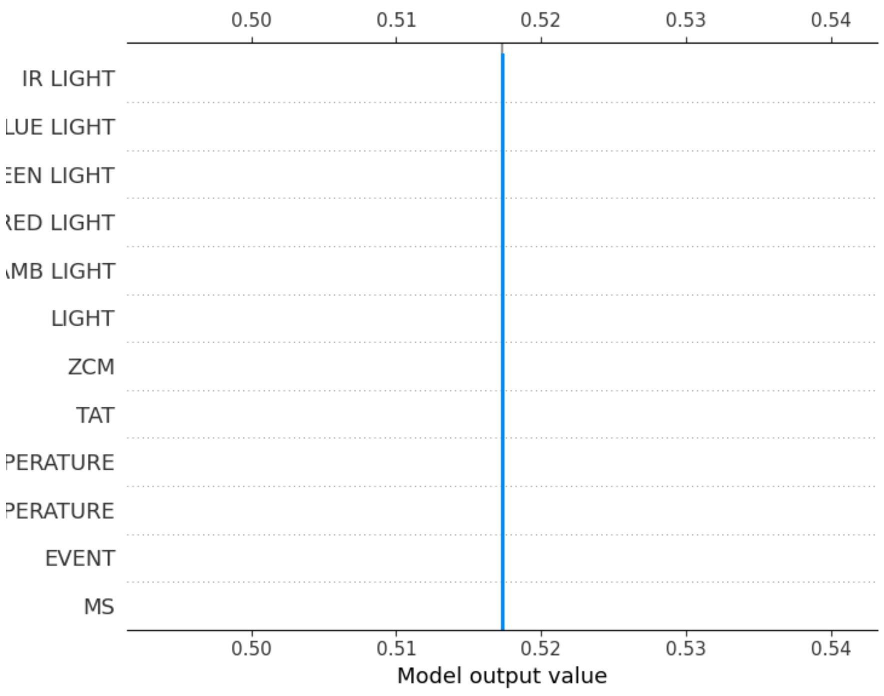


Figure 4.10: Decision Plot for patient 21.

But in other cases, as for patient five and patient twenty-five, we can see how the variables TAT, $REDLIGHT$, and ZCM performed an impact on the learning process of the model for this patient, as presented in Figures 4.11 and 4.12 for patient 5, and in Figures 4.13 and 4.14 for patient 25.

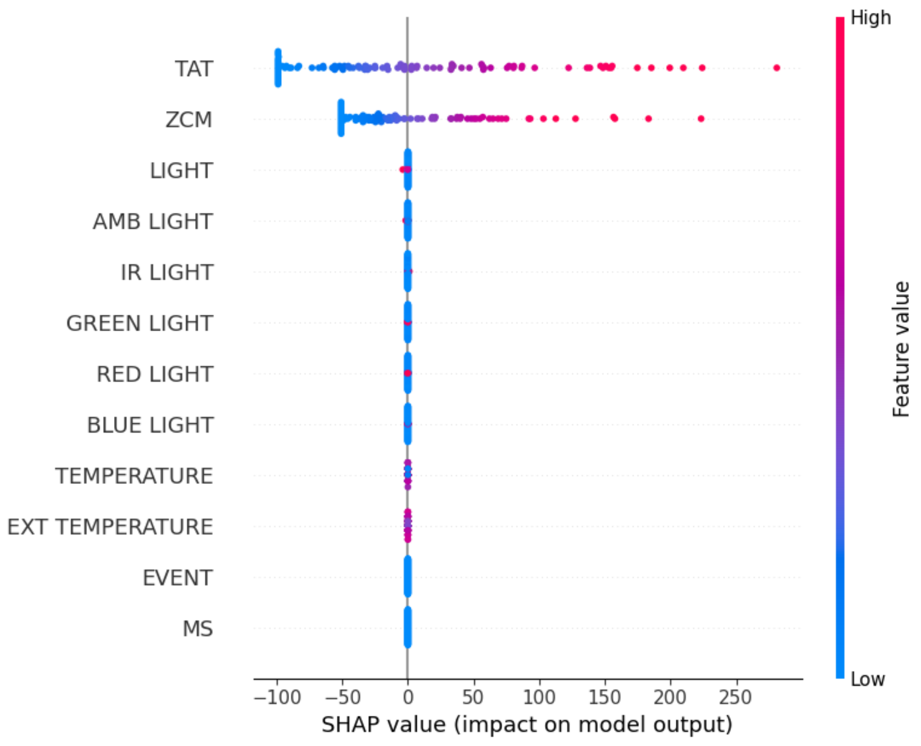


Figure 4.11: Summary Plot for patient 5.

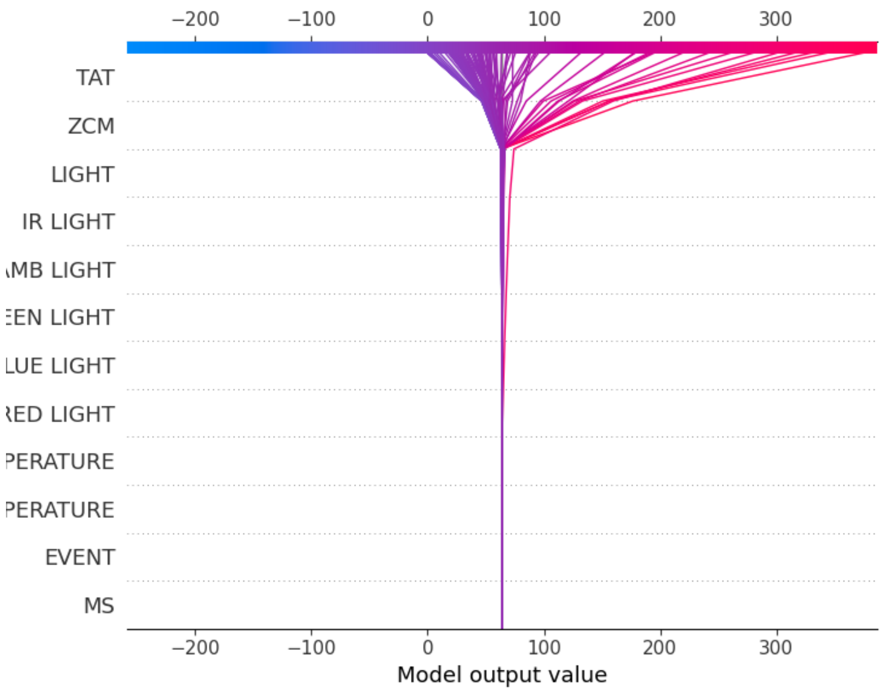


Figure 4.12: Decision Plot for patient 5.

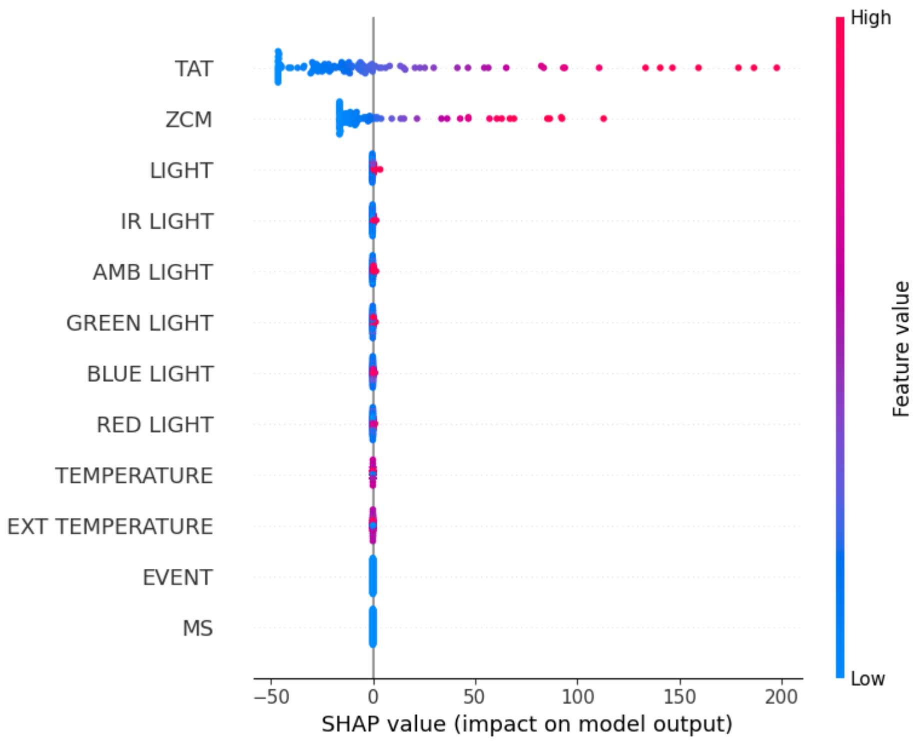


Figure 4.13: Summary Plot for patient 25.

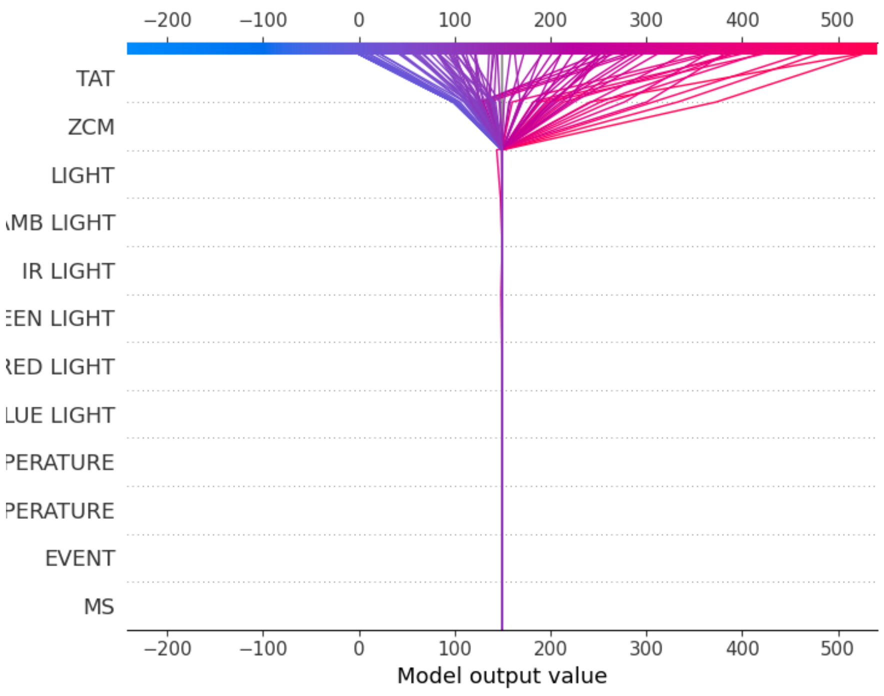


Figure 4.14: Decision Plot for patient 25.

In most cases where certain variables dominate others, TAT and ZCM consistently emerge as variables with a significant impact on prediction values. This is mainly due to their high correlation with the PIM, as both TAT and ZCM are indices of patient movement. Conversely, in other instances, luminosity-related variables such as $REDLIGHT$ and $LIGHT$ (a general combination of different light variables) significantly influence predictions. This indicates that, in some cases, luminosity may substantially impact the prediction more than variables like patient temperature or environmental temperature.

Chapter 5

Conclusion

The study aimed to evaluate various imputation methods for handling missing data in actigraphy measurements, explicitly focusing on Off-Wrist (OW) periods. The methods assessed included an Autoencoder with complete data, Temporal Moving Average Estimation (TMAE), Machine Learning (ML) models with M10 filtering, ML models using Dynamic Time Warping (DTW) with the Around Gap approach, and SHAP (SHapley Additive exPlanations) evaluations of ML models. Including SHAP and TMAE adds depth to the analysis by providing interpretability and a comparative baseline, respectively.

5.1 Autoencoder with Full Data

The Autoencoder demonstrated robust performance across most patients, evidenced by low MSE and MAE values and high R^2 scores nearing 1.00 (Table 4.2). This indicates the model's strong ability to reconstruct missing data accurately when complete data is available. The nparACT metrics further support this, showing minimal deviations between the original and reconstructed data (Table 4.4). The Autoencoder effectively preserved key circadian patterns, which is crucial for accurate actigraphy analysis.

The NaN value observed in the MASE metric for patient 21, although an exclusive case, is caused by a division by zero in the MASE calculation, highlighting sensitivity to data normalization that can result in constant time series. The Wilcoxon test results (Table 4.3) confirm the statistical significance of the Autoencoder's performance and the metrics the nparACT pointed out.

5.2 TMAE with Full Data

The TMAE approach provided an even more robust strategy for the autoencoders, adding Transformer layers. While it yielded low MSE and MAE values (Table 4.5), the R^2 values were significantly lower than the Autoencoder, indicating a less accurate fit to the data. The higher

MASE values above 1 suggest that TMAE performs worse than a naive benchmark, limiting its effectiveness for precise imputation tasks.

Although the Wilcoxon test results (Table 4.7) showed significant differences between the original and imputed data, the nparACT metrics (Table 4.8) revealed notable discrepancies in circadian rhythm parameters, indicating that TMAE may distort essential behavioral patterns. These findings suggest that while TMAE offers a computationally robust approach, it may not suit the actigraphy context.

5.3 ML Models with M10 Filtering

The ML models combined with M10 filtering demonstrated high performance, low error metrics, and high R^2 values (Table 4.9). Ensemble methods, particularly XGBoost, proved effective across most patients. The models maintained the integrity of crucial actigraphy metrics, as shown by the nparACT results (Table 4.11), indicating their capability to preserve circadian patterns during imputation.

However, the exceptionally high MASE values suggest that, despite the low MSE and high R^2, the models perform poorly compared to a naive benchmark. This inconsistency highlights a potential overfitting issue or sensitivity to specific data characteristics, which may not generalize well across different patients or datasets. The Wilcoxon test results (Table 4.10) confirmed statistical significance, but these factors may limit the models' practical effectiveness.

5.4 ML Models with DTW and Around Gap Approach

The integration of DTW with the Around Gap approach in ML models aimed to enhance the imputation by accounting for temporal dynamics and aligning similar patterns. The models achieved competitive performance, with low MSE and MAE values and high R^2 scores (Table 4.12). The nparACT metrics (Table 4.14) indicated that the reconstructed data closely matched the original regarding circadian rhythm parameters.

Nevertheless, variability in MASE values, particularly the significant outlier for patient 05, suggests that the method may be sensitive to individual differences in activity patterns. The Wilcoxon test results (Table 4.13) showed statistical significance and variability in T-statistics across patients. While the DTW approach can effectively capture temporal variations, it may require patient-specific adjustments or further refinement to improve imputation accuracy consistently.

5.5 SHAP Evaluations of ML Models

The application of SHAP provided insights into the interpretability of the ML models. For patients 05 and 25, SHAP analyses revealed that features like Total Activity Time (TAT), Red Light exposure, and Zero Crossing Mode (ZCM) significantly influenced the models' predictions (Figures 4.11–4.14). This interpretability may be necessary for understanding the model's decision-making process and identifying potential biases or areas for improvement.

In contrast, for patients 04 and 21, SHAP evaluations indicated that the features had a uniform impact on the predictions (Figures 4.7–4.10), suggesting that the models relied equally on all input variables or that the data lacked sufficient variability. This could point to overfitting or an inability of the model to discern patterns specific to these patients, emphasizing the need for tailored modeling approaches or feature engineering.

5.6 General Observations and Future Directions

The comparative analysis reveals that while advanced models like the Autoencoder and ML methods with DTW offer high imputation accuracy, they also present challenges such as potential overfitting, sensitivity to individual patient data, and computational complexity. Complex methods like TMAE, although less accurate, highlight the trade-off between computational efficiency and imputation precision.

The discrepancies in MASE values and the anomalies observed in certain patients underscore the importance of personalized approaches. Future work could focus on developing hybrid models that combine the strengths of different methods or incorporate adaptive algorithms that adjust to individual patient patterns.

Furthermore, increasing the number of patients beyond the nine in the study may be necessary to enhance the generalizability of data imputation models in actigraphy. A larger, more diverse sample would capture broader activity patterns and demographic variations, enabling models to account for differences in variables and lifestyles. This would reduce overfitting to a small, homogeneous group and ensure more robust, real-world applicability of imputation methods in clinical and research contexts.

An additional direction for future research could involve testing median models or state-space models as alternatives to traditional regressors for data imputation. Median models, such as SARIMA (Seasonal AutoRegressive Integrated Moving Average), offer robustness to outliers, computational efficiency, and the ability to handle seasonality in time-series data, making them well-suited for clinical and actigraphy datasets.

State-space models, such as Kalman filters and hidden Markov models (HMMs), could also be applied to the pipeline. These models excel in capturing temporal dependencies and managing stochasticity. Kalman Filters are ideal for continuous data with linear or nonlinear dynamics, while HMMs effectively model transitions between discrete states. Together, these methods provide a promising toolkit for enhancing imputation accuracy in clinical and time-series contexts.

Additionally, including SHAP analyses emphasizes model interpretability, which is essential for clinical applications where understanding the rationale behind predictions is as important as understanding the predictions themselves. Further research could explore integrating SHAP more deeply into the modeling process, such as retraining the models after identifying the most important features for that patient later in training to enhance accuracy and interoperability.

Finally, Federated Learning (FL) offers a novel approach to addressing actigraphy data loss by enabling collaborative model training across decentralized devices without sharing raw data. FL can leverage the distributed nature of actigraphy data collected from wearable devices, allowing each device to contribute to model updates while preserving user privacy. By incorporating FL, imputation models can learn from patterns across a wide range of devices, enhancing their ability to handle missing data effectively.

5.7 Conclusion

This work was able to propose a Pipeline to perform data imputation in actigraphy data. Both filtering and regression methods were developed and integrated, along with other necessary steps to fulfill the objective. The study demonstrates that imputation methods like the Autoencoder and ML models with advanced filtering techniques can effectively reconstruct missing actigraphy data, preserving critical circadian metrics. However, challenges such as overfitting, variability in patient performance, and the need for interpretability highlight the necessity for continued refinement. Incorporating interpretability tools like SHAP may offer a balanced approach for handling missing actigraphy data in diverse patient populations.

Bibliography

P. J. M. Ali. Investigating the impact of min-max data normalization on the regression performance of k-nearest neighbor with different similarity measurements. *ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY*, 10, 2022. ISSN 2410-9355. doi: 10.14500/aro.10955.

R. C. Anafi, R. Pellegrino, K. R. Shockley, M. Romer, S. Tufik, and A. I. Pack. Sleep is not just for the brain: Transcriptional responses to sleep in peripheral tissues. *BMC Genomics*, 14, 2013. ISSN 14712164. doi: 10.1186/1471-2164-14-362.

S. Ancoli-Israel, R. Cole, C. Alessi, M. Chambers, W. Moorcroft, and C. P. Pollak. The role of actigraphy in the study of sleep and circadian rhythms, 2003. ISSN 01618105.

L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach. Explaining anomalies detected by autoencoders using shapley additive explanations[formula presented]. *Expert Systems with Applications*, 186, 2021. ISSN 09574174. doi: 10.1016/j.eswa.2021.115736.

K. M. Asim, A. Idris, T. Iqbal, and F. Martínez-Álvarez. Earthquake prediction model using support vector regressor and hybrid neural networks. *PLoS ONE*, 13, 2018. ISSN 19326203. doi: 10.1371/journal.pone.0199004.

J. Baek, K. Han, and S. Choi-Kwon. Sleep diary- and actigraphy-derived sleep parameters of 8-hour fast-rotating shift work nurses: A prospective descriptive study. *International Journal of Nursing Studies*, 112, 2020. ISSN 00207489. doi: 10.1016/j.ijnurstu.2020.103719.

P. Bedi and P. Gole. Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network. *Artificial Intelligence in Agriculture*, 5, 2021. ISSN 25897217. doi: 10.1016/j.aiia.2021.05.002.

A. Brink-Kjaer, N. Gupta, E. Marin, J. Zitser, O. Sum-Ping, A. Hekmat, F. Bueno, A. Cahuas, J. Langston, P. Jennum, H. B. Sorensen, E. Mignot, and E. During. Ambulatory detection of isolated rapid-eye-movement sleep behavior disorder combining actigraphy and questionnaire. *Movement Disorders*, 38, 2023. ISSN 15318257. doi: 10.1002/mds.29249.

P. Chakri, S. Pratap, Lakshay, and S. K. Gouda. An exploratory data analysis approach for analyzing financial accounting data using machine learning. *Decision Analytics Journal*, 7, 2023. ISSN 27726622. doi: 10.1016/j.dajour.2023.100212.

J. Charest and M. A. Grandner. Sleep and athletic performance: Impacts on physical performance, mental performance, injury risk and recovery, and mental health, 2020. ISSN 15564088.

M. de Feijter, M. F. O'Connor, B. J. Arizmendi, M. A. Ikram, and A. I. Luik. The longitudinal association of actigraphy-estimated sleep with grief in middle-aged and elderly persons. *Journal of Psychiatric Research*, 137, 2021. ISSN 18791379. doi: 10.1016/j.jpsychires.2021.02.042.

M. Durbin, M. A. Wonders, M. Flaska, and A. T. Lintereur. K-nearest neighbors regression for the discrimination of gamma rays and neutrons in organic scintillators. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 987, 2021. ISSN 01689002. doi: 10.1016/j.nima.2020.164826.

D. Fekedulegn, M. E. Andrew, M. Shi, J. M. Violanti, S. Knox, and K. E. Innes. Actigraphy-based assessment of sleep parameters, 2020. ISSN 23987316.

E. M. C. Feliciano, S. L. Rifas-Shiman, M. Quante, S. Redline, E. Oken, and E. M. Taveras. Chronotype, social jet lag, and cardiometabolic risk factors in early adolescence. *JAMA Pediatrics*, 173, 2019. ISSN 21686203. doi: 10.1001/jamapediatrics.2019.3089.

L. Fiorillo, D. Pedroncelli, V. Agostini, P. Favaro, and F. D. Faraci. Multi-scored sleep databases: how to exploit the multiple-labels in automated sleep scoring. *Sleep*, 46, 2023. ISSN 15509109. doi: 10.1093/sleep/zsad028.

J. Garefelt, S. Gershagen, G. Kecklund, H. Westerlund, and L. G. Platts. How does work impact daily sleep quality? a within-individual study using actigraphy and self-reports over the retirement transition. *Journal of Sleep Research*, 31, 2022. ISSN 13652869. doi: 10.1111/jsr.13513.

M. A. Grandner and M. E. Rosenberger. *Actigraphic sleep tracking and wearables: Historical context, scientific applications and guidelines, limitations, and considerations for commercial sleep devices*. 2019. doi: 10.1016/B978-0-12-815373-4.00012-5.

S. Haghayegh, S. Khoshnevis, M. H. Smolensky, and K. R. Diller. Application of deep learning to improve sleep scoring of wrist actigraphy. *Sleep Medicine*, 74, 2020. ISSN 18785506. doi: 10.1016/j.sleep.2020.05.008.

Q. Han, B. Zheng, M. Cristea, and E. al. Trust in government regarding covid-19 and its associations with preventive health behaviour and prosocial behaviour during the pandemic: A cross-sectional and longitudinal study. *Psychological Medicine*, 53, 2023. ISSN 14698978. doi: 10.1017/S0033291721001306.

A. Ismail, S. Abdlerazek, and I. M. El-Henawy. Development of smart healthcare system based on speech recognition using support vector machine and dynamic time warping. *Sustainability (Switzerland)*, 12, 2020. ISSN 20711050. doi: 10.3390/su12062403.

J. H. Jang, J. Choi, H. W. Roh, S. J. Son, C. H. Hong, E. Y. Kim, T. Y. Kim, and D. Yoon. Deep learning approach for imputation of missing values in actigraphy data: Algorithm development study. *JMIR mHealth and uHealth*, 8, 2020. ISSN 22915222. doi: 10.2196/16113.

S. Jeganathan, A. R. Lakshminarayanan, N. Ramachandran, and G. B. Tunze. Predicting academic performance of immigrant students using xgboost regressor. *International Journal of Information Technology and Web Engineering*, 17, 2022. ISSN 15541053. doi: 10.4018/IJITWE.304052.

R. Jumabhoy, S. Maskevich, P. Dao, C. Anderson, J. Stout, and S. Drummond. Validation of consumer and research-grade activity monitors against polysomnography in healthy adults. *PsyArXiv Preprints*, pages 1–36, 2019. URL `https://osf.io/preprints/psyarxiv/mx2ae`.

C. Jyothsna, K. Srinivas, B. Bhargavi, A. E. Sravanth, A. T. Kumar, and J. N. Kumar. Health insurance premium prediction using xgboost regressor. In *Proceedings - International Conference on Applied Artificial Intelligence and Computing, ICAAIC 2022*, 2022. doi: 10.1109/ICAAIC53929.2022.9793258.

K. A. Kaplan, L. S. Talbot, J. Gruber, and A. G. Harvey. Evaluating sleep in bipolar disorder: Comparison between actigraphy, polysomnography, and sleep diary. *Bipolar Disorders*, 14, 2012. ISSN 13985647. doi: 10.1111/bdi.12021.

P. W. Khan and Y. C. Byun. Genetic algorithm based optimized feature engineering and hybrid machine learning for effective energy consumption prediction. *IEEE Access*, 8, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3034101.

J. Kim, J. Kong, and J. Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *Proceedings of Machine Learning Research*, volume 139, 2021.

H. M. Lehrer, Z. Yao, R. T. Krafty, M. A. Evans, D. J. Buysse, H. M. Kravitz, K. A. Matthews, E. B. Gold, S. D. Harlow, L. B. Samuelsson, and M. H. Hall. Comparing polysomnography, actigraphy, and sleep diary in the home environment: The study of women's health across the nation (swan) sleep study. *SLEEP Advances*, 3, 2022. ISSN 26325012. doi: 10.1093/sleepadvances/zpac001.

E. Lin, S. Mukherjee, and S. Kannan. A deep adversarial variational autoencoder model for dimensionality reduction in single-cell rna sequencing analysis. *BMC Bioinformatics*, 21, 2020. ISSN 14712105. doi: 10.1186/s12859-020-3401-5.

S. Mazza, H. Bastuji, and A. E. Rey. Objective and subjective assessments of sleep in children: Comparison of actigraphy, sleep diary completed by children and parents' estimation. *Frontiers in Psychiatry*, 11, 2020. ISSN 16640640. doi: 10.3389/fpsyt.2020.00495.

G. Mendonça Freire and M. Curi. Masked autoencoder transformer for missing data imputation of pisa. In A. M. Olney, I.-A. Chounta, Z. Liu, O. C. Santos, and I. I. Bittencourt, editors, *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, pages 364–372, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-64315-6.

J. A. Mitchell, M. Quante, S. Godbole, P. James, J. A. Hipp, C. R. Marinac, S. Mariani, E. M. C. Feliciano, K. Glanz, F. Laden, R. Wang, J. Weng, S. Redline, and J. Kerr. Variation in actigraphy-estimated rest-activity patterns by demographic factors. *Chronobiology International*, 34, 2017. ISSN 15256073. doi: 10.1080/07420528.2017.1337032.

E. Mosca, F. Szigeti, S. Tragianni, D. Gallagher, and G. Groh. Shap-based explanation methods: A review for nlp interpretability. In *Proceedings - International Conference on Computational Linguistics, COLING*, volume 29, 2022.

E. Nichols, J. D. Steinmetz, and E. al. Estimation of the global prevalence of dementia in 2019 and forecasted prevalence in 2050: an analysis for the global burden of disease study 2019. *The Lancet Public Health*, 7, 2022. ISSN 24682667. doi: 10.1016/S2468-2667(21)00249-8.

M. Niemeijer, M. D. Abràmoff, and B. V. Ginneken. Automated localization of the optic disc and the fovea. In *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS'08 - "Personalized Healthcare through Technology"*, 2008. doi: 10.1109/iembs.2008.4649969.

Y. Nohara, K. Matsumoto, H. Soejima, and N. Nakashima. Explanation of machine learning models using shapley additive explanation and application for real data in hospital. *Computer Methods and Programs in Biomedicine*, 214, 2022. ISSN 18727565. doi: 10.1016/j.cmpb.2021.106584.

R. Olu-Ajayi, H. Alaka, I. Sulaimon, F. Sunmola, and S. Ajayi. Building energy consumption prediction for residential buildings using deep learning and other machine learning techniques. *Journal of Building Engineering*, 45, 2022. ISSN 23527102. doi: 10.1016/j.jobe.2021.103406.

M. Oyeleye, T. Chen, S. Titarenko, and G. Antoniou. A predictive analysis of heart rates using machine learning techniques. *International Journal of Environmental Research and Public Health*, 19, 2022. ISSN 16604601. doi: 10.3390/ijerph19042417.

L. K. Pilz, M. A. D. Oliveira, E. G. Steibel, L. M. Policarpo, A. Carissimi, F. G. Carvalho, D. B. Constantino, A. C. Tonon, N. B. Xavier, R. D. R. Righi, and M. P. Hidalgo. Development and testing of methods for detecting off-wrist in actimetry recordings. *Sleep*, 45, 2022. ISSN 15509109. doi: 10.1093/sleep/zsac118.

E. M. Rogers, N. F. Banks, and N. D. Jenkins. The effects of sleep disruption on metabolism, hunger, and satiety, and the influence of psychosocial stress and exercise: A narrative review, 2024. ISSN 15207560.

P. Sai, M. Reddy, and J. P. Chandar. ”decision tree regressor compared with random forest regressor for house price prediction in mumbai”, 2023.

S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11, 2007. ISSN 15714128. doi: 10.3233/ida-2007-11508.

S. F. Schoch, S. Kurth, and H. Werner. Actigraphy in sleep research with infants and young children: Current practices and future benefits of standardized reporting, 2021. ISSN 13652869.

A. J. Scott, T. L. Webb, M. M.-S. James, G. Rowse, and S. Weich. Improving sleep quality leads to better mental health: A meta-analysis of randomised controlled trials, 2021. ISSN 15322955.

A. C. Tonon, L. K. Pilz, G. R. Amando, D. B. Constantino, R. B. Borges, A. Caye, F. Rohrsetzer, L. Souza, H. L. Fisher, B. A. Kohrt, V. Mondelli, C. Kieling, M. Idiart, A. Diez-Noguera, and M. P. Hidalgo. Handling missing data in rest-activity time series measured by actimetry. *Chronobiology International*, 39, 2022. ISSN 15256073. doi: 10.1080/07420528.2022.2051714.

H. Torabi, S. L. Mirtaheri, and S. Greco. Practical autoencoder based anomaly detection by using vector reconstruction error. *Cybersecurity*, 6, 2023. ISSN 25233246. doi: 10.1186/s42400-022-00134-9.

M. Tripathi. Facial image denoising using autoencoder and unet. *Heritage and Sustainable Development*, 3, 2021. ISSN 27120554. doi: 10.37868/hsd.v3i2.71.

S. Vikram, L. Li, and S. Russell. Handwriting and gestures in the air, recognizing on the fly. In *Conference on Human Factors in Computing Systems - Proceedings*, volume 2013-April, 2013. doi: 10.1145/2468356.2468567.

L. Weed, R. Lok, D. Chawra, and J. Zeitzer. The impact of missing data and imputation methods on the analysis of 24-hour activity patterns. *Clocks and Sleep*, 4, 2022. ISSN 26245175. doi: 10.3390/clockssleep4040039.

P. L. Yang, N. S. Chaytor, R. L. Burr, V. K. Kapur, S. M. McCurry, M. V. Vitiello, C. L. Hough, and E. C. Parsons. Rest-activity rhythm fragmentation and weaker circadian strength are associated with cognitive impairment in survivors of acute respiratory failure. *Biological Research for Nursing*, 25, 2023. ISSN 15524175. doi: 10.1177/10998004221109925.

W. Yu, M. Zhang, and Y. Shen. Spatial revising variational autoencoder-based feature extraction method for hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 59, 2021. ISSN 15580644. doi: 10.1109/TGRS.2020.2997835.

A. Zainab, A. Ghrayeb, M. Houchati, S. S. Refaat, and H. Abu-Rub. Performance evaluation of tree-based models for big data load forecasting using randomized hyperparameter tuning. In *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*, 2020. doi: 10.1109/BigData50022.2020.9378423.

M. D. Zambotti, N. Cellini, A. Goldstone, I. M. Colrain, and F. C. Baker. Wearable sleep technology in clinical and research settings. *Medicine and Science in Sports and Exercise*, 51, 2019. ISSN 15300315. doi: 10.1249/MSS.0000000000001947.

Q. Zhu, J. Chen, L. Zhu, X. Duan, and Y. Liu. Wind speed prediction with spatio-temporal correlation: A deep learning approach. *Energies*, 11, 2018. ISSN 19961073. doi: 10.3390/en11040705.