Transnational University Limburg is a unique collaboration of two universities in two countries: the University of Hasselt and Maastricht University.

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Faculty of Sciences

School for Information Technology

Master of Statistics and Data Science

Master's thesis

Unveiling the Digital Phenotype of Physical Activity Behavior in Community-Dwelling Older Adults

Anas Nazar Abdulghani

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Biostatistics

SUPERVISOR :
Prof. dr. Bruno BONNECHERE

2024
2025

UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculty of Sciences
School for Information Technology
Master of Statistics and Data Science

Master's thesis

Unveiling the Digital Phenotype of Physical Activity Behavior in Community-Dwelling Older Adults

Anas Nazar Abdulghani
Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Biostatistics

SUPERVISOR :
Prof. dr. Bruno BONNECHERE

Contents

1 Abstract

Background and motivation: Physical activity (PA) is an important factor for maintaining health and well-being, especially in older adults. Understanding patterns of PA can help in designing better interventions and monitoring strategies. With the increasing availability of wearable devices and mobile applications, detailed and continuous data on daily activity and related factors can be collected longitudinally. This thesis aims to apply machine learning methods to such data to predict PA patterns and identify key factors influencing these behaviors among community-dwelling older adults.

Objectives: The general aim of this thesis is to investigate the application of machine learning models in digital phenotyping with two main objectives. The two objectives are: (1) To identify important predictors of physical activity, mild depression status, and risk of fall using cross-sectional data. (2) To develop and evaluate predictive models for forecasting individual PA (step count) and determine the minimal window size required for accurate next-day PA predictions.

Materials and methods: The study utilized both cross-sectional and longitudinal datasets, integrating data from activity tracker devices and ecological momentary assessments (EMA). Cross-sectional analysis involved features obtained from questionnaires, physical tests, and self-reported variables to predict depression status, risk of fall, and PA levels using machine learning models like LightGBM, Elastic Net, and Linear or Logistic regression. Longitudinal analysis focused on forecasting step counts using time series data from wearable devices, employing models such as LightGBM, Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM).

Key findings: The most important predictors for the PA levels were items from the exercise self-efficacy scale (ESES) and exercise identity scale (EIS). In predicting fall risk, the key factor was the quadriceps score of the right leg. The primary predictor for mild depression status was a specific item from the International Physical Activity Questionnaire (IPAQ). Additionally, oxygen saturation (post-test) emerged as the most predictive variable when considering the IPAQ as a continuous measurement. In the longitudinal analysis, using a seven-day sequence of step count data provided the best performance for forecasting physical activity for the entire next day (comprising four time segments). In contrast, a six-day sequence was found to be optimal when predicting the number of steps for a single future time segment.

Limitations and future work:

Limitations of this thesis include reliance on selecting a single best model without leveraging stacking approaches, potential suboptimal temporal pattern learning by the LightGBM model, and limited hyperparameter tuning in the longitudinal analysis. Future work should explore advanced model tuning, stacking methods, and additional models that may better capture complex temporal dependencies. Also, it is recommended to collect more data by incorporating additional features and increasing the number of participants.

Conclusion: This thesis examined machine and deep learning models to address two objectives by using cross-sectional data to identify factors associated with PA levels in older adults, showing that self-efficacy was an important predictor. However, the overall prediction performance for PA and related outcomes was limited. In the longitudinal analysis, models were developed to predict future step counts using past activity data. It was found that a seven-day history of step counts provided the best next-day predictions, while features from EMA did not improve these predictions. Although some models were able to predict the step count accurately for some individuals, differences in activity patterns, methodological drawbacks, and the size of the dataset limited the ability to generalize the results for other participants. Further work with additional methods, larger and more diverse data is needed to improve model performance and support personalized health interventions.

2 Introduction

2.1 Background and motivation

2.1.1 Physical activity in older adults

According to the World Health Organization (WHO), the world population aged over 60 years will have doubled in number by 2050, with an estimated total of 2 billion people [1]. Aging is associated with some physiological changes, with reduced aerobic capacity (indicated by declining VO2max in inactive individuals) and sarcopenia (loss of skeletal muscle mass, strength, and function), which are crucial with respect to quality of life, functional independence, and mortality. These conditions can be exacerbated by physical inactivity [2]. In the broad definition of Physical activity (PA), it includes formal exercise, sports, and physical efforts performed as part of daily tasks, occupation, leisure, or active transportation [3].

On a global scale, physical inactivity, which is defined by the WHO as engaging in less than 150 to 300 minutes of moderate-intensity or 75 to 150 minutes of vigorous-intensity physical activity per week, remains prevalent in older adult populations. Specifically, 19–25% of individuals aged 60–69 years and 42–59% of those aged 80 years and older do not meet the PA guidelines for aerobic activity [4]. This can be associated with a rise in noncommunicable diseases such as cardiovascular disease, type 2 diabetes, stroke, and dementia [3].

Regular physical activity in older adults is associated with some health benefits, including improvements in physical function and enhanced mental and cognitive well-being [3]. Also, longitudinal studies suggest a reduction of risk of dementia, particularly Alzheimer's disease, for physically active individuals [2].

Furthermore, PA has a positive effect on functional independence in older adults, even for those individuals who are at risk of falls [3]. For example, structured exercise programs have been shown to have substantial positive effects on falls, functional ability, and overall capacity [4]. Moreover, multicomponent exercises can further improve these outcomes. [5].

To summarize, many studies have consistently concluded the beneficial effect of PA on health in older adults. It is estimated that 3.2 million deaths per year are due to physical inactivity. For this reason, sometimes PA is regarded as medicine for older adults [5].

2.1.2 Digital phenotyping

Digital phenotyping is an emerging approach to health data collection that uses digital tools like smartphones and wearables to passively and continuously monitor physiological, behavioral, and psychological metrics. By using this approach, researchers can build models over time for PA patterns [6].

According to a scoping review by Lee et al. [6], digital phenotyping has the potential for early intervention and prevention of serious medical conditions. This is particularly important for aging populations, who often struggle with recall bias when self-reporting PA. [6]. Daniels et al. [7] found that integrating ecological momentary assessment (EMA), wearable devices, and temporal frameworks strengthens the evaluation of PA. Additionally, their work indicated that low-intensity PA was influenced by motivation and self-efficacy, showing the importance of real-time contextual data in behavioral health assessments.

According to Song et al. [8], digital behavioral indicators like sleep behavior, PA, and heart rate variability can be considered as predictors for same-day and next-day depressive symptoms among socially at-risk older individuals who live in their usual environments. Furthermore, these technologies also support the daily individualized feedback on the health status of older individuals, which can enhance participation and contribute to positive health outcomes.

The clinical relevance of digital phenotyping stems from its alignment with the P4 medicine principles: Predictive, Preventive, Personalized, and Participatory care. This is useful in supporting early interventions in disease management, when conventional methods may be limited in detecting dynamic behavioral changes across diverse time and settings due to limited evaluations [9].

2.2 Importance of predicting physical activity

In recent years, machine learning–based predictive modeling has played a vital role in PA research by detecting activity levels, predicting adherence to PA goals, and producing individualized feedback, which are important to keep a sustained activity in aging populations [10] [11] . Deep learning- and machine learning-driven digital phenotyping methods offer promising new ways to capture within- and between-subject variation in physical activity, particularly when conventional methods like questionnaires are limited by recall bias or low temporal detail [12].

2.3 Ethical thinking, societal relevance, and stakeholder awareness

This thesis involves the analysis of existing datasets collected as part of ongoing research studies. The data used in the studies were anonymized before being shared with the author. Both the cross-sectional and longitudinal datasets were shared under ethical and institutional approval. The longitudinal data, which was collected using Garmin devices and the SEMA3 app, was approved by the Ethical Committee at Hasselt University.

This thesis aims to improve the understanding of physical activity behaviors in older adults, which can support the development of effective health interventions and policies to promote healthy aging. The findings may assist healthcare providers and policymakers in designing better strategies to encourage activity and prevent related health issues. Additionally, technology developers, such as companies developing the Garmin devices and the SEMA3 app, may benefit from the insights generated to enhance their products for more accurate monitoring and user engagement.

2.4 Research objectives for predicting physical activity in older adults

The general aim of this thesis is to explore how machine learning and deep learning models can be applied within the context of digital phenotyping to better understand and predict PA behaviors in older adults. Two distinct datasets are utilized for this aim: a cross-sectional dataset consisting of demographic, clinical, and psychological variables from older participants, and a longitudinal dataset combining step count data from wearable devices (Garmin) with EMA collected over two weeks.

- **Objective 1: The cross-sectional analysis**

  To identify baseline predictive factors of PA, risk of falling based on fall history in the past six months, and mild depression in a cross-sectional dataset of older adults using Logistic Regression, Linear Regression, regularized regression (Elastic Net), and tree-based gradient boosting (LightGBM). This objective focuses on between-subject variability in self-reported PA and its associations with demographic and other reported factors.

- **Objective 2: The longitudinal analysis**

  To develop time-series predictive models of step count using longitudinal Garmin wearable data, both alone and in combination with EMA variables. This objective leverages deep learning methods such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, and machine learning approaches such as Light Gradient-Boosting Machine (LightGBM) to explore within-subject temporal dynamics and assess whether contextual and psychological EMA inputs improve short-term PA predictions. In addition, this also aims to explore the minimal amount of period data (optimal time window) required to make reliable next-day predictions of PA.

3 Materials and Methods

3.1 Study Design and Participants

This thesis focuses on the analysis of data that were collected from the following study. The study used a two-week prospective observational design to gather detailed information on PA behaviors and their influencing factors. The study was registered at Clinical Trials.gov (NCT06094374) on 17 October 2023 and approved by the Ethical Committee of Hasselt University (B1152023000011). The full study protocol detailing recruitment strategies, data collection procedures, and analytical methods has been presented separately [13]. Informed consent was obtained from all subjects before participation. The cross-sectional part involved self-reported questionnaires to collect demographic and contextual data, as well as clinical tests to assess relevant health and functional status. Additionally, longitudinal data were collected through EMA and continuous monitoring using wearable devices. The study took place in a natural setting to ensure that participants could carry out their usual daily activities without disruption (ecological assessment). Participants were community-dwelling older adults aged 65 years and above, living independently either in their own homes or serviced apartments [7].

3.2 Data Description

3.2.1 Cross-sectional data

To collect the cross-sectional data, participants were asked to fill out questionnaires and also participated in a clinical evaluation. The questions encompassed various psychological and behavioral domains, including quality of life (WHOQOL), physical activity (IPAQ as a continuous measurement), depression (geriatric depression scale or GDS category), as well as sociodemographic information such as age, sex, marital status, and living situation.

Clinical measures included objective tests like the 6-minute walking distance test and body mass index (BMI). In addition to these, a comprehensive set of variables was collected encompassing lifestyle factors (e.g., smoking status, alcohol consumption, voluntary work), health indicators (e.g., blood pressure, heart rate, pain score, health score), mobility and physical capability measures (e.g., hand and leg muscle strength, balance tests), cognitive function tests (e.g., memory and reaction time scores), psychological scales (e.g., perceived stress scale (PSS), loneliness scale, goal attainment scale (GAS)), exercise motivation (e.g., exercise identity scale (EIS), exercise self-efficacy scale (ESES), behavioral regulation in exercise questionnaire (BREQ)), and digital health readiness (e.g., digital health readiness questionnaire (DHRQ) subscales). In total, 308 variables were systematically collected per participant, providing a rich multidimensional dataset capturing the physical, psychological, social, and contextual factors relevant to aging and digital phenotyping.

3.2.2 Longitudinal data

During the 14-day study period, participants' daily physical activity (step counts) was continuously recorded using the Garmin Vivosmart 5® activity tracker (Garmin International, Olathe, KS). Each participant had 56 time points (4 timesteps per day over 14 days), which corresponds to three-hour segments (e.g., 8:00–11:00, 12:00–15:00, 15:00–18:00, and 18:00–23:00). At each time segment or timestep, the number of steps was aggregated.

With regards to the EMA variables, participants used the SEMA3 smartphone application (Melbourne eResearch Group, Melbourne, Australia) and received four random prompts each day at times that were evenly distributed across the same four time intervals as for the PA recordings: 8:00–11:00, 12:00–15:00, 15:00–18:00, and 18:00–23:00.

At each prompt, participants were asked to rate five main areas: physical well-being, mental well-being, motivation, efficacy, and context. The assessments included questions about self-rated health, physical symptoms such as muscle stiffness, pain, dizziness, shortness of breath, and fatigue, as well as contextual factors and overall quality of life (QoL). To reduce response bias and improve data quality, the order of the questions was randomized [7].

3.3 Data preprocessing

3.3.1 Cross-sectional data

Variables were categorized based on their number of unique values. Specifically, variables with five or fewer unique values were treated as categorical, and they were dummy-coded before model training. In contrast, variables with six or more unique values were considered continuous and were treated as numerical predictors for model training.

Variables exhibiting very low or near-zero variance, characterized by having the same value in the majority of observations, were excluded from the analysis. This step was taken because such variables generally contribute little to predictive performance and can potentially create problems during model training [14].

All the cross-sectional analysis was done using `R` version 4.3.3.

3.3.2 Longitudinal data

The EMA and step count data were aligned using participant ID, date, and time segment. The resulting dataset captured within-subject temporal variation in physical activity and contextual or psychological conditions, with a focus on predicting the number of steps in the following day and finding the minimal time window for reliable predictions. In the longitudinal dataset, some participants had measurements for only a few days with large gaps between them, resulting in a high proportion of missing data. These participants were excluded from the analysis to ensure data completeness. Specifically, participants with more than 30% missing values in the outcome variable and without complete measurements over the 14-day period were removed. For those with more than 14 days of data, only the first 14 days were used to allow for a fair comparison. After applying these criteria, a total of 100 participants were included in the analysis.

The longitudinal analysis was conducted using `Python` version 3.10.18.

3.3.3 Missing data

To handle missing values in some features in the cross-sectional dataset, multiple imputations using the `mice` package in `R` were used. The method of imputation relied on the distribution of different variables. For categorical variables with more than two unique values, Proportional Odds Logistic Regression (polr) was used. Logistic Regression (logreg) was utilized to impute the binary variables, and Predictive Mean Matching (pmm) was used to impute the continuous variables. Ten imputations were performed with ten iterations to generate ten complete datasets.

For models that relied on the imputed datasets, such as Logistic Regression and Elastic Net, each complete dataset had its own coefficients, which were then used to generate the predictions on the test data, producing ten predicted values. These predictions were then averaged to obtain the final predicted value from the test set.

3.4 Predictive modeling for the cross-sectional data

3.4.1 Linear and logistic regression

Linear and Logistic Regression models were used to predict four outcomes in the cross-sectional dataset. Risk of fall, GDS category (mild depression status), and IPAQ category were binary outcomes, while IPAQ as a continuous measurement (IPAQ MET minutes/week) was a continuous outcome. Thus, linear regression was used for predicting the continuous outcome, while the binary outcomes were predicted using logistic regression models.

Linear Regression

Linear Regression is a statistical method used for predicting a continuous outcome. The general form of a multiple linear regression model, as formulated by [15], is:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_p X_{pi} + \varepsilon_i \qquad \text{(Eq. 1)}$$

For the i-th participant, Y_i represents the continuous measurement of IPAQ, $X_{1i}, X_{2i}, ..., X_{pi}$ are the predictors values for the i-th subject in the cross-sectional dataset, β_0 is the intercept and $\beta_1, ..., \beta_p$ are the coefficients for each predictor, and ε_i is the error term.

To estimate the coefficients, the least squares method was used, which minimizes the residual sum of squares (RSS):

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \qquad \text{(Eq. 2)}$$

This method yields a closed-form solution for $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)^T$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \qquad \text{(Eq. 3)}$$

where $\hat{\boldsymbol{\beta}}$ is the vector of estimated coeffcients, $\mathbf{X}$ is the design matrix and $\mathbf{y}$ is the vector of outcomes [15].

Logistic Regression

Logistic Regression is used for binary classification problems, where the outcome is binary (takes the values of 0 for failure and 1 for success). In the cross-sectional dataset, three outcomes of risk of fall, GDS category, and IPAQ category were modeled using Logistic Regression. Logistic regression models use log-odds of success vs failure as outcome, and use the logit link function, and they are formulated by [15] as:

$$\log\left(\frac{P(Y_i = 1 \mid X_i)}{1 - P(Y_i = 1 \mid X_i)}\right) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_p X_{pi} \qquad \text{(Eq. 4)}$$

which gives the logistic function:

$$P(Y_i = 1 \mid X_i) = \frac{e^{(\beta_0+\beta_1 X_{1i}+\beta_2 X_{2i}+\cdots+\beta_p X_{pi})}}{1 + e^{(\beta_0+\beta_1 X_{1i}+\beta_2 X_{2i}+\cdots+\beta_p X_{pi})}} \qquad \text{(Eq. 5)}$$

Where $P(Y_i = 1 \mid X_i)$ is the probability of success for the i-th subject. Model coefficients are estimated using maximum likelihood estimation (MLE), which looks for the set of parameters $\boldsymbol{\beta}$ that maximizes the likelihood of observing the data. The likelihood for n independent observations is:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \left(\frac{1}{1 + e^{-X_i^T \boldsymbol{\beta}}}\right)^{y_i} \left(1 - \frac{1}{1 + e^{-X_i^T \boldsymbol{\beta}}}\right)^{1-y_i} \qquad \text{(Eq. 6)}$$

This is solved using an iterative optimization algorithm like Newton-Raphson.

Given the large number of predictors, the top 10 to 30 predictors were selected based on information gain for fitting the models.

As for variables with high pairwise correlations of 60% or more, only one was selected while the others were excluded from the analysis.

3.4.2 Elastic Net

The Elastic Net is a regularization and variable selection technique that can overcome some of the challenges encountered by traditional penalized regression methods, especially in high-dimensional settings where the number of predictors p exceeds the number of observations n. This method is suited for datasets like the cross-sectional data, which consists of 108 observations and 308 predictors. Since many of these predictors are likely to be highly correlated, the Elastic Net is an appropriate method to address this issue.

Elastic Net was developed to do both shrinkage and automatic variable selection, combining the advantages of LASSO and ridge regression. LASSO uses an ℓ_1-norm penalty to support sparsity by setting some coefficients exactly equal to zero, while ridge regression uses an ℓ_2-norm penalty to shrink the size of all coefficients, particularly for predictors with high correlation. Following the formulation by Hastie et al. [16], the Elastic Net's objective function for Linear Regression can be expressed as:

$$(\hat{\beta}_0, \hat{\beta}) = \arg\min_{\beta_0, \beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - X_i^\top \beta)^2 + \lambda \left((1-\alpha)\frac{1}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1 \right) \right\} \qquad \text{(Eq. 7)}$$

where:

- n: The number of samples or participants in the cross-sectional data.
- β_0: The model intercept.
- β: The estimated vector of regression coefficients of the predictors.
- $X_i^\top$: The p-dimentional vector representing the predictors' values for the i-th sample.
- y_i: The observed continuous outcome for the i-th individual.
- λ: The regularization parameter that controls the overall degree of penalty. $\lambda \geq 0$.
- α: The mixing parameter:
  - $\alpha = 1$: corresponds to LASSO (pure ℓ_1 regularization).
  - $\alpha = 0$: corresponds to ridge regression (pure ℓ_2 regularization).
  - $0 < \alpha < 1$: corresponds to Elastic Net.
- $||\beta||_1$: The ℓ_1 norm of the vector of coefficients β, defined as $\sum_{j=1}^{p} |\beta_j|$. This supports sparsity by shrinking some coefficients exactly to zero.
- $\|\beta\|_2^2$: The squared L2 norm of β, defined as $\sum_{j=1}^{p} \beta_j^2$. This promotes small but nonzero values of the coefficients to stabilize the model in the presence of multicollinearity [16].

The regularization parameter λ and the mixing parameter α were optimized through cross-validation to select the values that minimize prediction errors.

The Elastic Net was used for regression (for the continuous measurement of IPAQ) and classification (GDS category, risk of fall, and IPAQ category). For the latter, the previous framework can be extended to Generalized Linear Models (GLMs) by replacing the residual sum of squares with a negative log-likelihood function as formulated by Hastie et al. [16]:

$$(\hat{\beta}_0, \hat{\beta}) = \arg\min_{\beta_0,\beta} \left\{ -\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \beta_0 + X_i^T\beta\right) + \lambda\left((1-\alpha)\frac{1}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1\right) \right\} \qquad \text{(Eq. 8)}$$

where y_i is the observed categorical outcome for the i-th participant, and $\ell\left(y_i, \beta_0 + X_i^T\beta\right)$ is the log-likelihood term for subject i.

3.4.3 Light Gradient Boosting

Light Gradient Boosting (LightGBM, also abbreviated as LGBM) is a gradient boosting framework that uses tree-based learning algorithms designed for efficient training, particularly suitable for complex structured data, such as the cross-sectional dataset.

LightGBM builds an ensemble of decision trees sequentially, where each new tree is added to correct the residuals or errors made by the previous trees. According to [15], the general formula for the boosting method is:

$$\hat{f}(x) = \sum_{b=1}^{B} r f_b(x). \qquad \text{(Eq. 9)}$$

where $\hat{f}(x)$ is the predicted value of the b-th tree, B is the total number of trees, and r is the learning rate that regulates the learning process of the model.

Unlike other gradient boosting methods, such as Extreme Gradient Boosting (XGBoost), LightGBM employs a leaf-wise tree growth strategy with depth constraints, which often leads to improved performance [17].

Given the presence of features with missing values in the cross-sectional dataset, LightGBM uses a sparsity-aware split algorithm. It learns the directions for missing values, and it utilizes them without imputation during the building of trees.

To help with the classification and the regression problem in the cross-sectional dataset, LightGBM was chosen alongside Elastic Net due to its ability to capture complex relationships between the predictors and the outcomes.

There are several parameters that need to be tuned for the LightGBM (LGBM) model:

- learning_rate (learn_rate): Controls the rate r at which the model learns.
- n_estimators (trees): The number of trees (boosting rounds) B to build.
- max_depth (tree_depth) The maximum depth of a tree.
- min_child_samples (min_n) The minimum number of data points needed to create a leaf.
- min_split_gain (loss_reduction) Minimum loss reduction needed to make a split at a tree node.
- subsample (sample_size) The subsampling rate, which is the fraction of the training data sampled for each tree.
- reg_alpha (lambda_l1) L1 regularization applied to leaf weights to promote sparsity..
- reg_lambda (lambda_l2) L2 regularization applied to leaf weights to help decrease model complexity.
- num_leaves (num_leaves) The maximum number of leaves permitted in a tree.

3.4.4 Metrics for the cross-sectional data analysis

The cross-sectional dataset was split into a train (70%) and a test (30%) set using a stratified splitting approach. Stratification splitting ensures that the class distribution in each set is similar to that in the complete dataset. This may avoid bias that can arise in the estimation of the performance if one class is under- or over-represented in either set. Next, stratified k-fold cross-validation (CV) on the training set for model hyperparameter tuning and selection was performed. This is done to keep the class distribution similar in each fold. In K-fold CV, k-1 folds are used for training, and the remaining fold (hold-out set) is used for validation. This ensures that every sample is used for both training and validation. It also reduces overfitting and makes the model generalize better to new unseen samples [18]. Stratified splitting and stratified CV help to preserve the class distribution throughout the process of training and validation, which improves the generalizability of the model to unseen new data.

To calculate the CV for a metric during training, $\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{X}_i$, where X can be recall, specificity, precision, Precision-Recall Area Under the Curve (PR AUC), etc.

Hyperparameter tuning

Bayesian optimization is used since it is more efficient than the full grid search approach, and it typically offers better optimized parameters than random search. The method involves treating the performance of a model as an unknown function that needs to be optimized. It constructs a probabilistic model (Gaussian process) to predict better settings or combinations of parameter values based on previous observations. The model takes into account uncertainty and also focuses on exploiting more promising areas in the parameter space. At each step, it chooses the next set of parameters by maximizing a criterion (e.g., expected improvement) by using previous information to make better choices [19].

Model comparisons

Three different models were fitted separately for the binary and continuous outcomes. This approach allowed for the comparison of model performance using various evaluation metrics to determine the model with the best prediction performance.

A variety of metrics were used that were selected based on the distribution of each outcome variable. These served to assess the performance of the models and compare different models. For the IPAQ as a continuous measurement, Mean Absolute Error (MAE) was one of the metrics used. MAE is calculated as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \qquad \text{(Eq. 10)}$$

where it measures the absolute difference between the predicted value $\hat{y_i}$ and the true observed value y_i, and taking the average of them yields MAE.

Median Absolute Error (MedAE) was also used to evaluate the regression models, which can be calculated as follows:

$$\text{MedAE} = \text{median}\left(\,|y_i - \hat{y}_i|\,\right) \qquad \text{(Eq. 11)}$$

where $i = 1, ..., n$. MedAE is a better metric to use than MAE for the evaluation of models when an outcome is skewed, since it is less sensitive to outlying observations [20].

For binary classification, each prediction can fall into one of four categories when it is compared to the true value or label. A true positive (TP) is when the model correctly predicts a positive outcome, while a true negative (TN) occurs when a negative outcome is predicted correctly. In contrast, a false positive (FP) occurs when a model falsely predicts a positive value for a negative label, and a false negative (FN) occurs when a positive label is incorrectly classified as negative. These four categories help to calculate the performance metrics for binary classifications. A 2x2 confusion matrix is shown in table 1, which can provide a good way for measuring the prediction performance, where the diagonal entries represent the correct prediction (TP and TN), while the off-diagonal elements show the number of misclassifications made by the model (FP and FN).

The metrics that were used in the classification, as formulated by [21]:

- Recall (Sensitivity) = $\frac{TP}{TP+FN}$
- Specificity = $\frac{\text{TN}}{\text{TN+FP}}$

- Precision = $\frac{TP}{TP+FP}$
- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Balanced accuracy = $\frac{\text{Sensitivity+Specificity}}{2}$
- F1 score = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}}$
- Area Under Precision-Recall Curve (PR AUC): The Area Under the Precision-Recall Curve (PR AUC) is calculated by measuring the area under the curve that plots precision against recall across all possible classification thresholds. PR AUC can provide a more informative assessment of model performance when dealing with imbalanced datasets. In such cases, PR AUC is often preferred over the Area Under the Receiver Operating Characteristic Curve (ROC AUC), because ROC AUC can be misleading by giving an overly optimistic evaluation when the model misclassifies most of the minority class instances [22]. Therefore, PR AUC was used as the primary evaluation metric for selecting the best classification model.

	Truth	
Prediction	**Yes (positive)**	**No (negative)**
Yes (positive)	TP	FP
No (negative)	FN	TN

Table 1: Structure of a confusion matrix used in binary classification

Class imbalance

Class imbalance can negatively impact model training by reducing the ability to identify minority classes accurately. To address this, class weights were applied during training to assign higher importance to minority class observations and improve model performance.

3.5 Modeling for the longitudinal data

3.5.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of Artificial Neural Networks (ANNs) developed to model sequential data, making them suitable for forecasting physical activity in sequential data. In contrast to feedforward neural networks, RNNs use information from previous time steps, creating a memory of past input that helps the network to learn temporal dependencies.

The formulations used in the following description of the RNN architecture follow the ones presented in [23].

RNN architecture

In a simple RNN, input data is introduced into the network model sequentially, being processed one timestep at a time. To compute the current hidden state h_t at time t, the network takes an input vector x_t and combines it with the previous hidden state h_{t-1} from the previous time step. h_t and the output y_t are computed as follows:

$$h_t = f\left(W_i^h(\mathbf{x}_t + b_i) + W_h^h(h_{t-1} + b_h)\right) \quad \text{(Eq. 12)}$$

$$y_t = g\left(W_h^o(h_t + b_o)\right) \quad \text{(Eq. 13)}$$

where W_i^h, W_h^h, and W_h^o are the input, recurrent, and output weight matrices. b_i, b_h, and b_o are the respective bias vectors. $f(\cdot)$ is an activation function (e.g., ReLU). $g(\cdot)$ is often a linear transformation for regression [23].

Through this recursive procedure, RNNs can capture the dependencies between different time steps in a sequence.

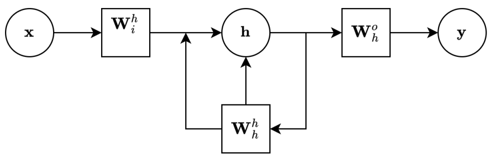

Figure 1: Simple structure of RNN

Figure 1 shows the architecture of an RNN. There are three primary layers: the input layer, the hidden layer, and the output layer.

- Input layer: $\mathbf{x}$ represents the input data at the current time step t, which could be the number of steps, or the EMA variables after normalization.
- Hidden layer: the input passes through the input weight matrix $\mathbf{W}_i^h$, which projects it into the hidden state. At the same time, the recurrent weight matrix $\mathbf{W}_h^h$ is multiplied with the previous hidden state $\mathbf{h}[t-1]$. They are then combined with the bias terms and introduced to an activation function such as *ReLU*, to get the current hidden state $\mathbf{h}[t]$. This enables the network to retain information from previous steps.
- Output layer: The output weight matrix $\mathbf{W}_h^o$ transforms the current hidden state $\mathbf{h}[t]$ to produce the output $\mathbf{y}[t]$. This output can be the predicted number of steps.

This structure allows the RNN to learn sequential patterns in longitudinal data. The same parameter weights (and biases) are used at each time step to make the model generalize across a variety of temporal positions.

However, a simple RNN struggles to learn long-term dependencies due to the vanishing gradient problem. For this purpose, recurrent layers such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are used to address this issue.

LSTM

LSTM networks are a type of RNN that were developed to address some limitations that were encountered in simple RNNs in capturing long-term dependencies. They do not suffer from vanishing gradient during training, since they have gated methods that enable the model to retain or discard information throughout long sequences, which improves memory control [23].

- $f_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)$

- $i_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)$
- $o_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)$
- $\tilde{\mathbf{C}}_t = g_1(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)$
- $\mathbf{C}_t = (f_t \times \mathbf{C}_t + i_t \times \tilde{\mathbf{C}}_t)$
- $\mathbf{h}_t = g_2(\mathbf{C}_t) \times o_t$

where $\mathbf{x}_t$ is the input vector at time t. f_t, i_t, and o_t are forget, input, and output gates, respectively. $\sigma(\cdot)$ is a sigmoid activation function ($\sigma(x) = \frac{1}{1+e^{-x}}$), $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_c$, $\mathbf{W}_o$, $\mathbf{U}_f$, $\mathbf{U}_i$, $\mathbf{U}_c$, and $\mathbf{U}_o$ are weight matrices. $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_c$, and $\mathbf{b}_o$ are bias vectors. $g_1(\cdot)$ and $g_2(\cdot)$ are non-linear activation functions.

Each component in the cells has a unique role in regulating the information flow. The forget gate controls how much information to remove from the previous state $\mathbf{C}_{t-1}$. The input gate regulates the amount of influence that the new candidate $\tilde{\mathbf{C}}_t$ should have on the new current state $\mathbf{C}_t$. To generate the hidden state $\mathbf{h}_t$, LSTM applies a nonlinear transformation to the current state and filters it by using the output gate, which controls what information is passed next.

GRU

GRU is a variant of the LSTM that models the temporal dependencies in sequential data. Unlike LSTM, GRU merges the forget and input gates into a single update gate, which regulates the amount of information to forget or remember. As a result, it has fewer parameters (weights) to estimate, making its training faster than the LSTM architecture. The update gate regulates how much information in the cell should be updated by the candidate state. Additionally, there is a reset gate that controls how much the previous state should influence the current state.

- $z_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z)$
- $r_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r)$
- $\tilde{\mathbf{h}_t} = g(\mathbf{W}_h\mathbf{x}_t + r_t \times \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h)$
- $\mathbf{h}_t = (1 - z_t) \times \mathbf{h}_{t-1} + z_t \times \tilde{\mathbf{h}_t}$

where z_t is the update gate and r_t is the reset gate. $\mathbf{W}_z$, $\mathbf{W}_r$, $\mathbf{W}_h$, $\mathbf{U}_z$, $\mathbf{U}_r$, and $\mathbf{U}_h$ are weight matrices. $\mathbf{b}_z$, $\mathbf{b}_r$, and $\mathbf{b}_h$ are the bias terms. $\tilde{\mathbf{h}_t}$ and $\mathbf{h}_t$ are the candidate state and the hidden state, respectively. Figure 2 shows the cells of both LSTM and GRU networks.

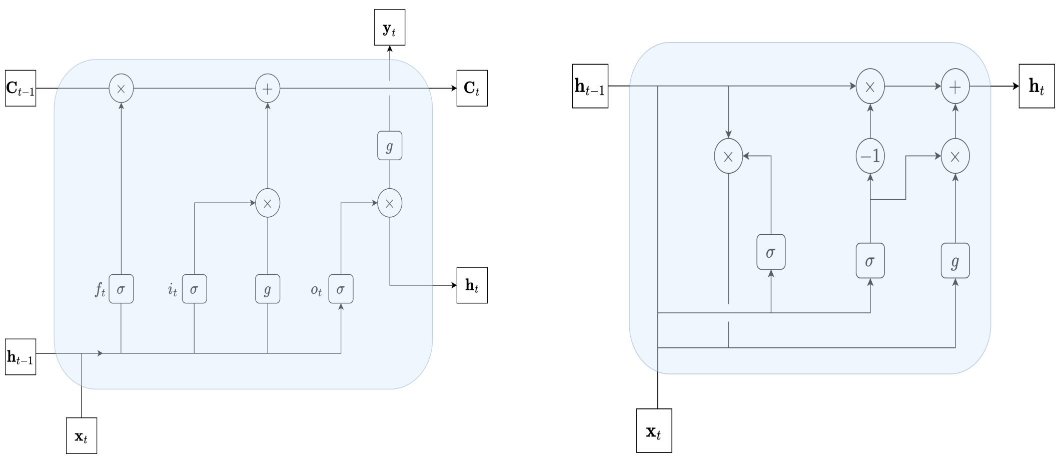

Figure 2: Architectural comparison of LSTM (left) and GRU (right) cells

3.5.2 LightGBM for time series forecasting

LightGBM was applied to time series forecasting in the longitudinal dataset by using lagged features as inputs. This approach has been shown to be valid for forecasting, provided that appropriate feature engineering is performed [24].

Lagged feature construction for LightGBM forecasting

To predict the number of steps at a given time t, lagged versions of the outcome variable were constructed as features from previous time steps. For example, if the current time is t_4, then the model uses the values at t_3, t_2, and t_1 as input features.

Table 2: Example of lagged feature construction (single timestep)

Time	Steps (Number of steps)	Lag 1	Lag 2	Lag 3
t_1	1200	—	—	—
t_2	1500	1200	—	—
t_3	1350	1500	1200	—
t_4	1700	1350	1500	1200
t_5	1600	1700	1350	1500

Other longitudinal predictors, such as the EMA variables (e.g., motivation, physical well-being), were lagged similarly.

3.5.3 Training and parameter estimation:

The participants in the longitudinal dataset were randomly divided into training (70%), validation (10%), and testing (20%) sets.

To train the model to forecast the continuous outcome of step count, the predicted values are compared with the actual or target values, which helps to construct a loss function. The main parameters (weights and biases) are estimated by minimizing the Mean Absolute Error (MAE) loss function:

$$\text{MAE}(y, y^*) = \frac{1}{T}\sum_{t=1}^{T} |y_t - y_t^*| \tag{Eq. 14}$$

Where T is the sequence length, y_t^* is the target value at time t, and y_t is the predicted value of the step count.

The MedAE was used as an evaluation metric because it is less sensitive to outlying observations compared to other evaluation metrics:

$$\text{MedAE}(y, y^*) = \underset{|y_t - y_t^*|}{\text{median}} \tag{Eq. 15}$$

where $t = 1, ..., T$

Model comparison

To determine the minimum number of days needed as input to predict the physical activity for the following day (consisting of 4 timesteps), the predictive performance of several model configurations for forecasting step count was compared. In total, 6 combinations were tested: LSTM with EMA variables (LSTM Steps + EMA), LSTM without EMA features (LSTM Steps only), GRU with EMA variables (GRU Steps + EMA), GRU without EMA variables (GRU Steps only), LightGBM with EMA features (LGBM Steps + EMA), and LightGBM using only lagged features derived from the step count variable (LGBM Steps only).

The primary metric used to evaluate the models was the MedAE divided by the median of the test data (MedAE/Median). This metric is scale-invariant because it accounts for the scale of the data, and lower values indicate better model performance.

Backpropagation Through Time

To train the RNN and update the values of weights, gradients of the loss functions with respect to the parameters are computed using Backpropagation Through Time (BPTT). In this method, the network is unrolled over time, and propagation is performed across the time steps.

The model parameters were optimized using the Adam optimizer, which is an adaptive learning method that is based on first-order and second-order moments. One advantage of Adam is that it adaptively adjusts the learning rate for each parameter, and this often leads to better performance [23].

Success criterion

Model evaluation was performed by forecasting short-term PA, measured as the number of steps at the next time point, based on a lagged sequence of previous activity. For each participant, the predictions were assessed using the percentage error, calculated as:

$$\text{Percentage error at time } i = \begin{cases} \frac{|\hat{y}_i - y_i|}{y_i}, & \text{if } y_i \neq 0 \\ \frac{|\hat{y}_i - y_i|}{1}, & \text{if } y_i = 0 \end{cases}$$

where $\hat{y}_i$ is the predicted value at timestep i and y_i is the actual value at the same timestep. If the actual value is zero, the denominator is set to 1 to avoid division by zero. A single prediction at a timestep i is considered correct if this percentage error is less than or equal to 0.10. A successful prediction for a particular participant is then defined as having at least 0.80 of their predicted values with percentage errors of 0.10 or less.

3.6 Outcome transformation:

To improve the training and performance of the regression models, the outcome variable in the longitudinal analysis was transformed using the Yeo-Johnson transformation, which helps to reduce skewness in highly skewed data [25]. This transformed outcome was used during the model training process. After obtaining predictions from the models, the values were converted back to the original scale by applying the inverse transformation, using the parameter λ optimized from the training data.

The Yeo-Johnson transformation of a continuous outcome (y) is:

$$\psi(\lambda, y) = \begin{cases} \frac{(y+1)^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0,\ y \geq 0 \\ \log(y+1) & \text{if } \lambda = 0,\ y \geq 0 \\ \frac{-[(-y+1)^{2-\lambda} - 1]}{2-\lambda} & \text{if } \lambda \neq 2,\ y < 0 \\ -\log(-y+1) & \text{if } \lambda = 2,\ y < 0 \end{cases} \quad \text{(Eq. 16)}$$

4 Results

4.1 Cross-sectional Analysis

4.1.1 Exploration

Table 3 shows summary statistics of some of the continuous variables according to their distribution, including the mean, standard deviation (SD), median, and 25th and 75th percentiles in the cross-sectional dataset. The mean age of the participants was 70.1 years (SD = 4.59), and the median BMI was 26.3 (23; 28.4). For physical activity as a continuous measurement, participants reported a median activity of 5143.50 MET-minutes/week (2642; 9973.3).

The table also shows the summary of categorical variables. The majority of the participants were married (72.2%), they were living with a partner (78.7%), and most of them were retired (97.2%).

Regarding the categorical outcome variables, according to the IPAQ categorization, 71.3% of the participants were highly active, while only one participant was categorized as having low physical activity levels. Due to the insufficient representation of the low activity group, the single participant in this category was excluded from the analysis. Consequently, the classification task was adjusted to a binary problem using only the moderate (as the negative class) and high activity (as the positive class) categories, as a single sample is insufficient for effective model training. Furthermore, 16.7% of participants experienced a fall incidence in the past 6 months, and 33.3% had mild depression according to GDS.

Table 3: Cross-sectional data summary statistics. Continuous data are presented as mean (SD) or median (p25; p75) according to the distribution of the data. The outcome variables are in bold.

Continuous variable	Statistic	Minimum - Maximum
Age (years)	70.1 (4.59)	64–87
BMI (kg/m^2)	26.3 (23.0; 28.4)	19–42.3
6min walking distance test	572.4 (90.8)	240–855
Speed	5.91 (0.80)	3.8–8.4
WHOQOL Physical Health	76.0 (11.8)	39.29–100
WHOQOL Psychological	72.3 (10.2)	45.83–91.67
WHOQOL Social	75.0 (66.7; 83.3)	25–100
WHOQOL Environment	83.7 (10.1)	56.25–100
IPAQ MET-min/week	5143.5 (2642.0; 9973.3)	99–64848
Categorical variable	value	n (%)
Sex	male	47 (44.52%)
	female	60 (55.56%)
	other	1 (0.93)
Marital state	Single	8 (7.4%)
	Living together	9 (8.3%)
	Married	78 (72.2%)
	Divorced	8 (7.4%)
	Widow	5 (4.6%)
Physical constraints	Yes	8 (7.4%)
	No	100 (92.6%)
Retired	Yes	105 (97.2%)
	No	3 (2.8%)
Living situation	Living with partner	85 (78.7%)
	Living alone	20 (18.5%)
	Living with children	1 (0.9%)
	Other	2 (1.9%)
IPAQ category	Low	1 (0.9%)
	Moderate	30 (27.8%)
	High	77 (71.3%)
GDS category	Mild depressed	36 (33.3%)
	Not depressed	72 (66.7%)
Falling in the past 6 months	yes	18 (16.7%)
	No	90 (83.3%)

4.1.2 Metrics

Table 4 presents the performance comparison of the models for mild depression status prediction. The LightGBM model achieved the highest PR AUC of 0.8, outperforming the PR AUC of

Logistic Regression and Elastic Net models.

The LightGBM classification model achieved a recall (sensitivity) of 0.545, indicating a moderate ability to correctly identify positive cases, while its specificity of 0.818 reflects a strong performance in correctly identifying negative cases. The model's precision was 0.6, suggesting a reasonable proportion of true positive predictions among all positive predictions. Overall, the F1 score of 0.571 balances precision and recall, and the balanced accuracy of 0.682

The LightGBM model achieved a PR AUC of 0.381 in predicting fall risk, indicating limited overall ability to distinguish minority cases. The recall (sensitivity) was 0.333, showing that the model correctly identified only a third of actual fall risk cases, highlighting challenges in detecting the positive class. The specificity was 0.750, reflecting a relatively good performance in correctly identifying individuals without fall risk. Precision was 0.222, meaning that among those predicted as at risk, only about one-fifth were true positives, indicating a high false positive rate. The F1 score was 0.267, reflecting the balance between precision and recall. The balanced accuracy was 0.542, representing the average of recall and specificity, and indicating moderate classification performance due to class imbalance.

With regards to the classification task distinguishing between high and moderate levels of PA based on the IPAQ category, the LightGBM model achieved a PR AUC score of 0.809, demonstrating a strong ability to discriminate between classes across different thresholds compared to other models. The model's recall was 0.875, indicating that it successfully identified a high proportion of individuals with high physical activity. Precision was 0.808, showing that most of the predicted high activity cases were correct and showing reliable positive predictions. The F1 score was 0.840, indicating a good balance between precision and recall. Specificity was 0.444, suggesting the model had difficulty in correctly identifying the moderate activity class. The balanced accuracy was 0.660, reflecting overall moderate accuracy that accounts for both sensitivity and specificity in the presence of class imbalance.

The models' performance in predicting IPAQ MET minutes per week was assessed using multiple error metrics in table 5, with a focus on the MedAE divided by the median (MedAE/Median) of the observed values of the test data. The LightGBM model achieved the lowest MedAE/Median value of 0.551, indicating the best prediction performance among the models. In comparison, the LR and EN models exhibited higher MedAE/Median values of 0.859 and 0.785, respectively. While LightGBM provides better prediction of IPAQ MET minutes per week compared to the other two models, the overall prediction error remains substantial, reflecting the challenges of modeling PA using the cross-sectional data.

	Truth	
Prediction	**Yes**	**No**
Yes	6	4
No	5	18

GDS LGBM

	Truth	
Prediction	**Yes**	**No**
Yes	2	7
No	4	21

Falling LGBM

	Truth	
Prediction	**Yes**	**No**
Yes	21	5
No	3	4

IPAQ LGBM

Table 6: Confusion matrices for the selected models (Yes = positive class, No = negative class).

Table 4: Evaluation metrics for binary outcomes (LR = Logistic Regression, EN = Elastic Net, LGBM = LightGBM).

Metric	GDS			Fall			IPAQ		
	LR	EN	**LGBM**	LR	EN	**LGBM**	LR	**EN**	LGBM
F1 Score	0.615	0.476	0.571	0.353	0.300	0.267	0.303	0.682	0.840
Precision	0.533	0.500	0.600	0.273	0.214	0.222	0.556	0.750	0.808
Recall (Sensitivity)	0.727	0.455	0.545	0.500	0.500	0.333	0.208	0.625	0.875
Specificity	0.682	0.773	0.818	0.714	0.607	0.750	0.556	0.444	0.444
Accuracy	0.697	0.667	0.727	0.676	0.588	0.676	0.303	0.576	0.758
Bal. Accuracy	0.705	0.614	0.682	0.607	0.554	0.542	0.382	0.535	0.660
PR_AUC	0.444	0.504	0.800	0.174	0.190	0.381	0.653	0.764	0.809

Table 5: Evaluation metrics for IPAQ MET minutes/week (LR = Linear Regression, EN = Elastic Net, LGBM = LightGBM).

Model	MAE	MedAE	MAE/Mean	MedAE/Median
LR	7349	4439	0.801	0.859
EN	6049	3974	0.704	0.785
LGBM	2788	6102	0.711	0.551

4.1.3 Predictive factors

Figure 3 shows the most important predictors for several outcome variables based on the best-performing models selected from the previous analyses. The LightGBM variable importance scores were based on gain, which represents the percentage contribution of each feature to the model, calculated from the total gain of the splits involving that feature.

For the GDS category, the LightGBM model highlighted an item from the IPAQ as the most important predictive factor. The second and third most important predictors were quadriceps strength on the left side and BMI, respectively. As for the prediction of risk of fall using the LightGBM model, the most predictive feature was the quadriceps strength of the right leg. Regarding the IPAQ category prediction with the LightGBM model, the three most important predictive variables were an item from ESES, an item from the EIS, and the 6-minute walking distance test. Moving to the IPAQ as a continuous measurement, the primary predictive variable was oxygen saturation (post-test), followed by an item from the WHOQOL questionnaire, and the EIS total score.

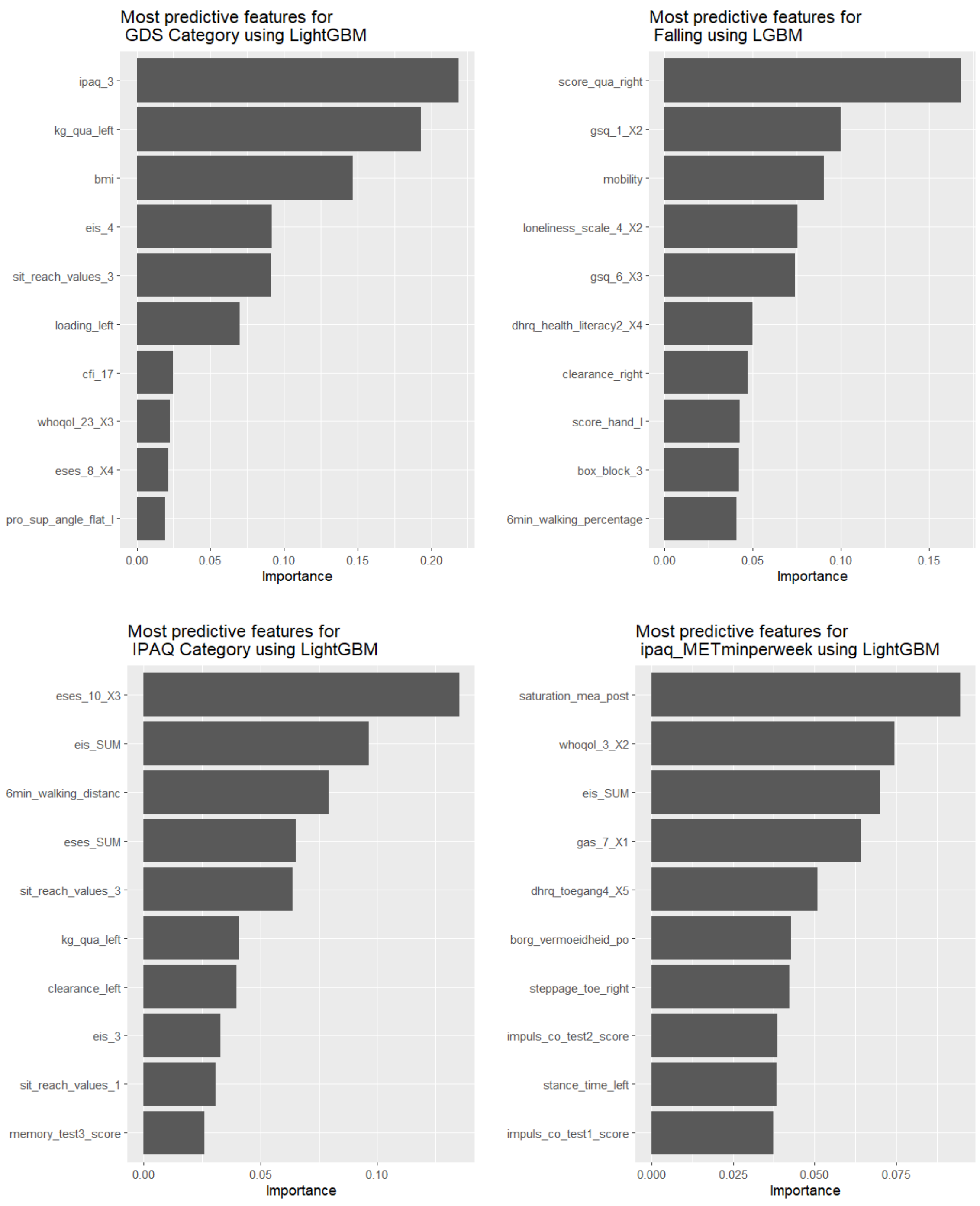

Figure 3: Most important features for the cross-sectional selected models

4.2 Longitudinal Analysis

4.2.1 Exploration

Table 7 summarizes the variables of the integrated dataset from the Garmin device and SEMA3 app. The results are obtained after processing the data. The main outcome of interest is the number of steps (Steps), with a median of 1143 steps per time period (p25 = 375, p75 = 2374), and it ranges between 0 and 21459 steps. The distribution of the number of steps is strongly right-skewed, with a large number of zero values and fewer observations with high step counts, as shown in Figure 4.

The EMA variables collected from the SEMA3 app had a range from 0 to 100. Physical well-being had a median of 23.81 (14.3; 33.3). Similarly, mental well-being was low with a median of 23.81 (14.3; 42.9). Both of these features had a right-skewed distribution, as shown in Figure 4. In contrast, motivation to be active had a median of 85.71 (57.1; 100). The median of the average Self-efficacy level was 100. Finally, a median context of 92.86 suggests that most participants were in environments that were supportive of physical activity. The distributions of motivation, self-efficacy, and context variables are left-skewed, as illustrated in Figure 4.

After the datasets were combined and properly aligned, the percentage of missing step count data measured by the Garmin device was 2.8%, while each of the EMA variables had a missingness of 62.2%.

Table 7: Longitudinal variables summary statistics

Variable	**Median (p25; p75)**	**Minimum - Maximum**	**Missing (%)**
Steps	1143 (375; 2374)	0–21459	157 (2.8%)
Physical	23.8 (14.3; 33.3)	9.52–100	3483 (62.2%)
Mental	23.8 (14.3; 42.9)	14.29–100	3483 (62.2%)
Motivation	85.7 (57.1; 100)	3.57–100	3483 (62.2%)
Efficacy	100.0 (71.4; 100)	14.29–100	3483 (62.2%)
Context	92.9 (71.4; 100)	14.29–100	3483 (62.2%)

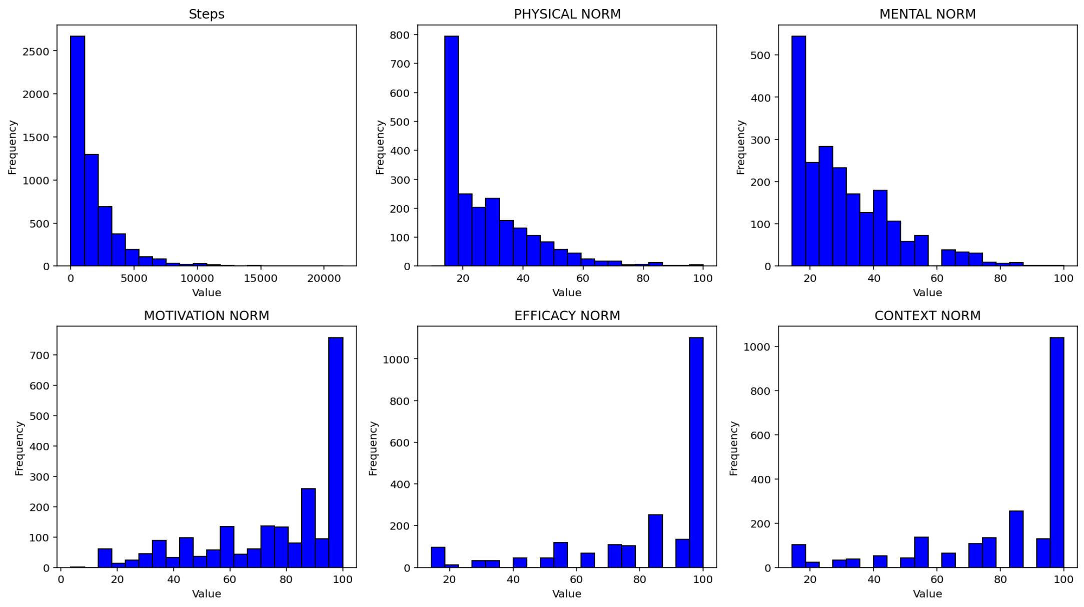

Figure 4: Histograms of longitudinal variables

Figure 5 shows the longitudinal step count data for four selected participants, representing different patterns observed across the study duration. The plots indicate considerable variation within participant 76, whose step counts ranged from 0 to over 7500 and changed substantially over time. Participants 73 and 89 exhibited distinct step count patterns characterized by sharp increases, indicating occasional periods of elevated physical activity. On the other hand, participant 6 had a smaller range of step counts, mostly below 3,000, showing less variation in their step counts. These patterns can also highlight notable between-subject differences in physical activity levels.

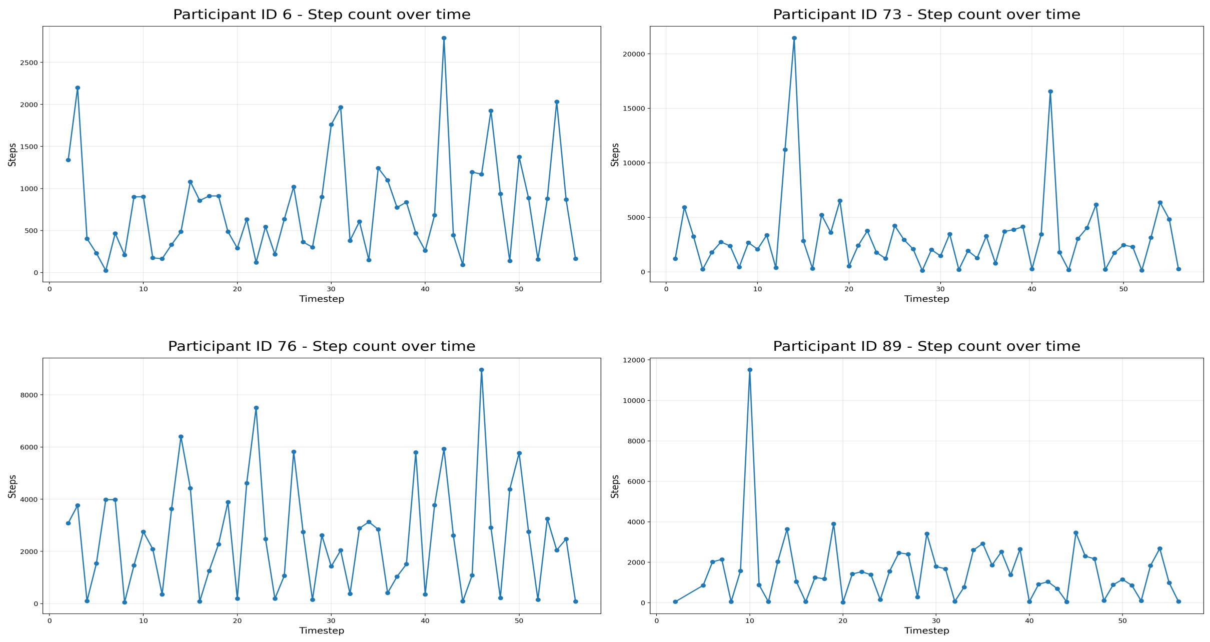

Figure 5: Selected plots for participants' longitudinal profiles

Figure 6 displays the distribution of step count (Steps) before and after applying the transformation. The transformed values show considerably less skewness compared to the original data.

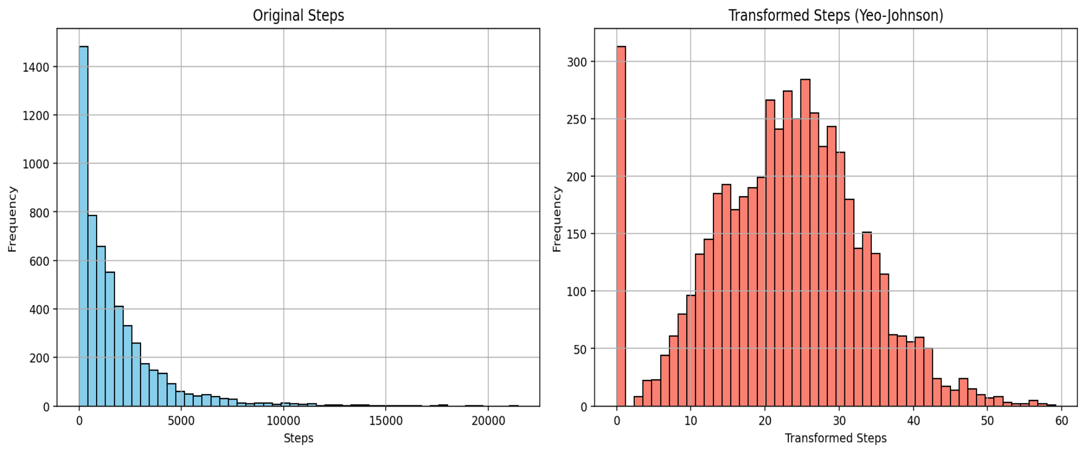

Figure 6: Outcome transformation using Yeo-Johnson transformation

4.2.2 Model specifications

Training of the GRU and LSTM models was conducted using 20 epochs, a batch size of 16, and a learning rate of 0.005. The model architectures consisted of the following layers:

Table 8: GRU and LSTM model specifications

GRU model	LSTM model
Masking layer for missing values	Masking layer for missing values
GRU (128 units, return sequences)	LSTM (128 units, return sequences)
GRU (64 units, no return sequences)	LSTM (64 units, no return sequences)
Dense (16 units, ReLU activation)	Dense (16 units, ReLU activation)
Dense (1 unit, output layer) for single-step prediction	Dense (1 unit, output layer) for single-step prediction
Dense (4 units, output layer) for multi-step (4 timesteps) prediction	Dense (4 units, output layer) for multi-step (4 timesteps) prediction

The parameter values applied in the LightGBM models are summarized in table 9

Table 9: LightGBM parameters

Parameter	Value
`n_estimators`	3000
`num_leaves`	1000
`max_depth`	100
`min_child_samples`	1
`min_split_gain`	0
`subsample`	1
`learning_rate`	0.005
`reg_alpha`	0.01
`reg_lambda`	0.01

4.2.3 Model comparisons

Figure 7 shows the model comparisons to predict the number of steps for the entire next day (four timesteps). The blue line represents the baseline performance (common sense model), which predicts the next step count by simply using the current step count. This approach does not involve any modeling and is included only as a reference point for comparing the performance of the developed models. All six models outperformed this baseline.

The results of the model comparisons indicate that the LightGBM model without EMA input (LGBM (Steps only)) achieved the best performance, with the lowest MedAE/median error ratios across days two to seven. Its error decreased gradually over the seven days, reaching a minimum of 0.31 on day seven. The LightGBM model with EMA features demonstrated worse performance, with MedAE/median ratios between 0.41 and 0.48 over seven days, showing no improvement from adding the lagged EMA features to the input.

The GRU model using only previous steps as input showed moderate performance, with error values ranging from approximately 0.44 to 0.52.

As for the GRU model with EMA features, it exhibited higher overall errors, mostly exceeding 0.6 and reaching up to 0.72 on day six.

The errors for LSTM (Steps + EMA) and GRU (Steps + EMA) were close across the days, indicating comparable predictive ability between these two model types. As for the LSTM (Steps only) model, it showed fluctuation in error ratios across the days compared to the GRU (Steps only) model.

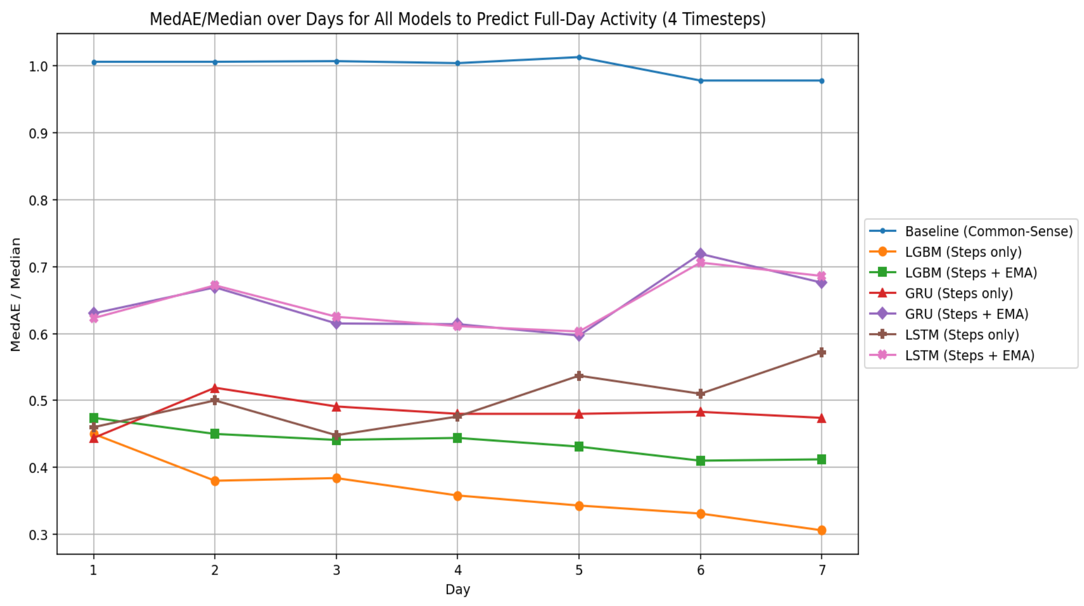

Figure 7: Median Absolute Error (MedAE) / Median over days for different models predicting next-day step counts (four timesteps), lower values indicate better models' performance

The LightGBM model, using step counts from the past seven days, was selected to forecast PA for the next four timesteps. Table 10 shows the performance of these LightGBM (Steps only) for forecasting a full day, along with the mean and median step count in the test data. The evaluation on the test set resulted in a MedAE of 414.37 steps and a MedAE/Median ratio of 0.306.

Table 10: LightGBM (Steps only) model evaluation metrics on the test set for forecasting a full day PA

Metric	**Value**
MAE	981.15
MedAE	**414.37**
Mean	2083.78
Median	1355.00
MAE / Mean	0.471
MedAE / Median	**0.306**

Figure 8 shows the model comparisons to predict the following number of steps for a single timestep only. The results showed that the LightGBM model using only previous step counts consistently achieved low MedAE/median error ratios between 0.26 on day six and 0.31 on day three, maintaining stable performance across the days and showing low sensitivity to input sequence length. In comparison, the LightGBM model with EMA features had higher error values, ranging from 0.40 to 0.45. The GRU model using step counts only exhibited error values from approximately 0.42 to 0.53, while the GRU model with EMA included had errors between 0.55 and 0.76. The LSTM (Steps only) model showed decreased errors on day one and day six (about 0.46) compared to the other days. As for the LSTM with EMA model, it reached a peak error of approximately 0.65 on day seven, while the LSTM (Steps only) model presented lower error rates compared to the LSTM with EMA, with error ratios close to those of the GRU (Steps only) model.

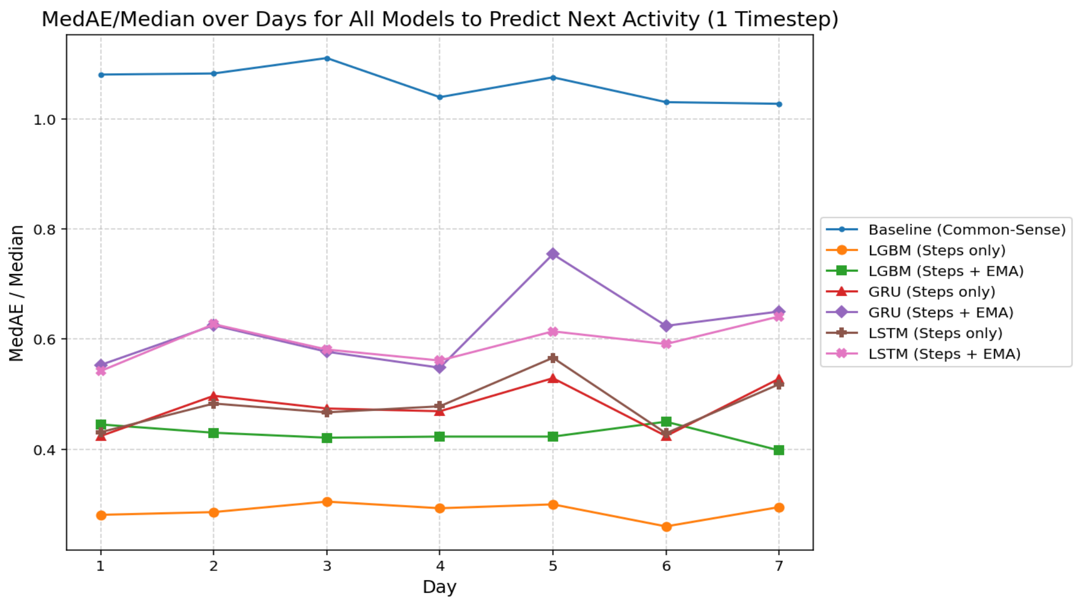

Figure 8: Median Absolute Error (MedAE) / Median over days for different models predicting next-timestep PA (a single timestep), lower values indicate better models' performance

Table 11 shows the metrics of the selected mode for forecasting a single timestep with 6 days of input. The model achieved a MedAE of 345.93 steps and a MedAE/Median ratio of 0.260.

Table 11: LightGBM (Steps only) model evaluation metrics on the test set for single timestep forecasting

Metric	**Value**
MAE	933.57
MedAE	**345.93**
Mean	2041.32
Median	1330.00
MAE / Mean	0.457
MedAE / Median	**0.260**

To further examine the behavior of the models, an additional analysis was performed using a fixed sequence length of six days, with different temporal arrangements of inputs and targets. Instead of using sequences covering the entire day, each input consisted of step counts from the same time segment (e.g., morning, noon, afternoon, or evening) across six consecutive days. The target was either the step count for the same time segment on the following day (e.g., using six mornings to predict the next morning) or the step count for a different time segment on the same or next day (e.g., using six afternoons to predict the next noon). This approach was intended to investigate whether certain time-of-day combinations provide more predictive information for step count and to compare model performance when predicting within the same time segment versus across different segment configurations. The results of these models are presented in Figure 9.

The LightGBM model trained solely on lagged step count features achieved the lowest MedAE/median values overall. Specifically, the afternoon-to-afternoon prediction, using a sequence of six step counts from the afternoon to predict the number of steps in the next afternoon, had a MedAE/Median error ratio of 0.27. The noon-to-noon and morning-to-morning predictions each showed error ratios of 0.36, while the evening-to-evening prediction had a ratio of 0.31. These findings suggest that the model performed best for forecasting the afternoon PA.

When using morning segments as input, the prediction errors were 0.32 for predicting noon PA, 0.33 for afternoon, and 0.38 for evening activity. Predicting the morning PA from the previous afternoon step counts had a relatively high error ratio of 0.48. In contrast, a lower error of 0.34 was obtained by predicting evening PA from afternoon input. Using evening PA as input achieved high error ratios of 0.49 and 0.50 for predicting morning and noon step counts. In contrast, it yielded lower error ratios of 0.30 and 0.31 for predicting afternoon and evening PA, respectively.

The top-right heatmap shows the LightGBM model results when EMA variables were incorporated alongside lagged step count inputs. Compared to the model without EMA variables, the inclusion of EMA features resulted in higher MedAE/median error ratios across most time segment combinations, indicating a modest decline in predictive performance. The afternoon-to-afternoon prediction showed the lowest error ratio of 0.35.

The two heatmaps in the middle show the results for the GRU models. In the GRU (Steps only) model, the overall MedAE/median error ratios are higher compared to those of the LightGBM models for morning-to-morning and afternoon-to-afternoon configurations. The best performance was observed when using evening input to predict evening (0.41), as well as predicting afternoon PA from morning input (0.41).

When EMA variables were added to the GRU model, as illustrated in the middle heatmap on the right, the highest error ratios continued to occur when predicting morning PA from all time segments, similar to the pattern seen in the GRU model using the previous steps only. In contrast, the afternoon-to-afternoon predictions exhibited the lowest error ratio of 0.39, comparable to the pattern observed in the LightGBM models.

The heatmaps at the bottom show the performance of the LSTM models with and without EMA data. For the LSTM model using only the previous step counts, the best performance was observed when predicting the afternoon segment from the afternoon input, with an error ratio of 0.36. This result surpassed both GRU models for the same time segment configuration. After adding EMA data to the LSTM model (right heatmap), the prediction error ratios for the noon target generally decreased.

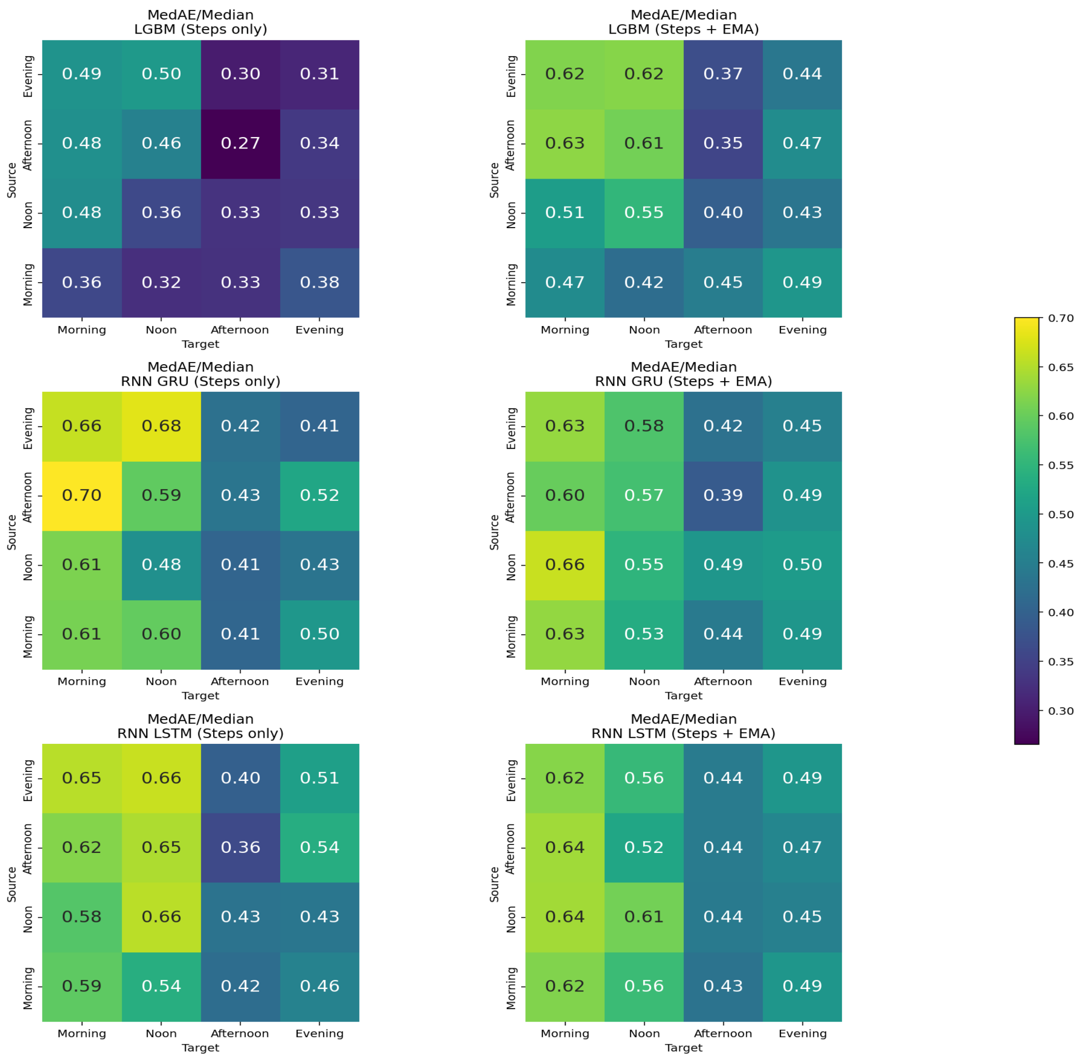

Figure 9: MedAE/Median across time segments and models.

Table 12 summarizes the evaluation of the LightGBM (Steps only) model's predictions of the number of steps at the next time point (single timestep), for individual participants.

Predictions were based on a six-day lagged sequence of previous PA. For each participant, the table reports the total number of predictions, the number of correct predictions (defined as having a percentage error of 10% or less at a timestep), and the percentage of correct predictions out of the total number of predictions within the participant.

Among the 20 participants in the test set, four participants satisfied this success criterion. Their respective correct prediction rates were notably high, ranging from 93.10% to 100%, suggesting that the model was capable of capturing meaningful temporal patterns in these individuals' physical activity behavior. For example, participant 61 had 32 out of 32 predictions classified as correct (100%), reflecting exceptional model performance for this individual.

In contrast, the majority of participants (16 out of 20) fell below the 80% threshold. For some individuals, the percentage of correct predictions was extremely low (6.25% for Participant 59), indicating substantial model underperformance and showing that the model failed to generalize effectively for these participants.

Table 12: Per-participant accuracy summary based on the proportion of predictions with percentage error $\leq 10\%$. Participants with at least 80% accurate predictions are highlighted in bold.

Participant ID	**Total predictions**	**Correct predictions**	**Percentage correct predictions**
2	32	10	31.25%
3	32	7	21.88%
14	32	9	28.13%
25	32	6	18.75%
38	32	20	62.50%
44	32	7	21.88%
52	32	10	31.25%
56	29	27	**93.10%**
59	32	2	6.25%
61	32	32	**100.00%**
63	31	9	29.03%
69	31	8	25.81%
70	32	8	25.00%
73	32	9	28.13%
74	32	16	50.00%
80	32	10	31.25%
93	32	8	25.00%
109	32	32	**100.00%**
111	32	32	**100.00%**
112	31	9	29.03%

Furthermore, a Leave-One-participant-Out (LOO) was conducted using six-day input to predict the next single step count using the LightGBM model without EMA. The testing procedure involved iteratively holding out the data from one participant as the test set, while training the model on the data from the other 99 participants using the parameters in table 9. This process was repeated for each participant in the whole dataset, so that every individual's data was used once as a test set. The error was calculated separately for each participant's prediction, based on the model trained without their data. Figure 10 shows the per-participant success rates for the following single-step count predictions using the LightGBM model without EMA inputs and a six-day input sequence. Out of the 100 participants, only 43 of them met the success criterion (in green bars).

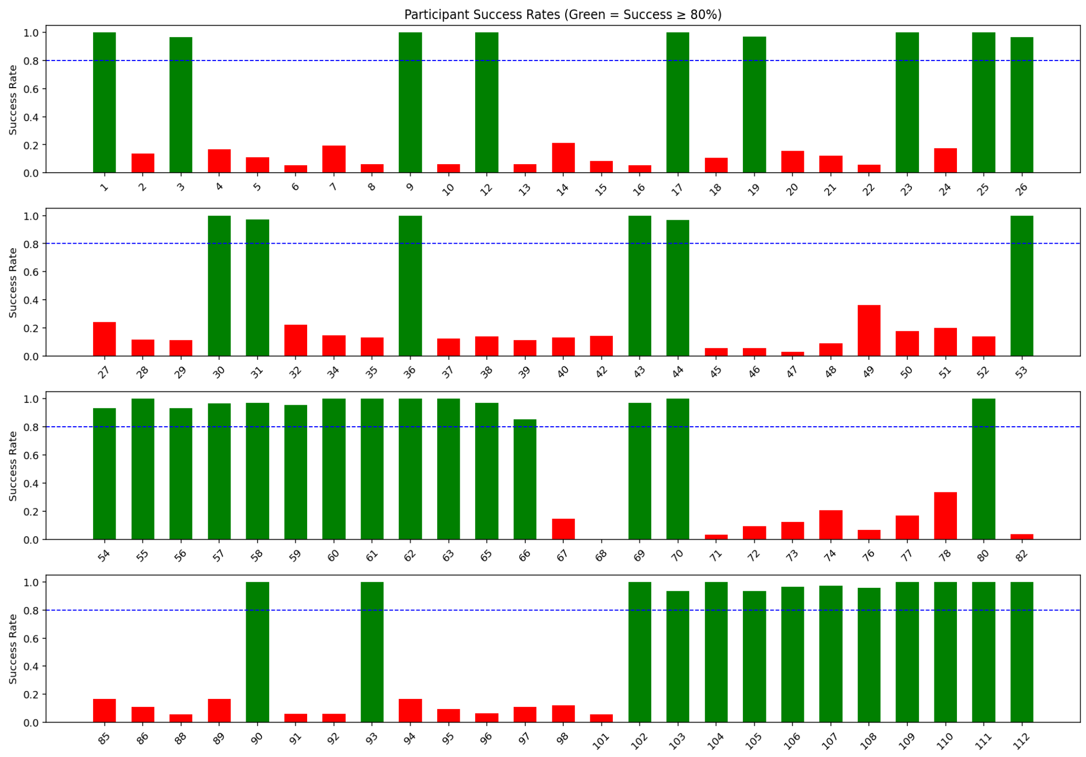

Figure 10: Per-participant success rates, defined as the proportion of predictions with percentage error $\leq 10\%$. Each bar represents an individual participant. Green bars indicate participants who met the predefined success criterion (success rate $\geq 80\%$), while red bars indicate participants who did not meet this threshold.

Table 13 presents the p-values from different tests, including the Wilcoxon rank-sum test for age and IPAQ as continuous measurements, and Fisher's exact test for the other variables, conducted to assess whether there was a systematic difference between participants in meeting the success criterion. No variables were statistically significant, indicating no evidence of systematic differences based on the measured characteristics.

Table 13: P-values from Wilcoxon and Fisher's exact tests examining differences in participant characteristics between those meeting and not meeting the success criterion.

Variable	**p-value**
Age	0.8026
IPAQ category	0.3657
Sex	1
Fall risk	0.5984
GDS category	0.6683
IPAQ MET-min/week	0.549

Figure 11 presents the predicted and actual step counts for 4 participants using the LightGBM model without including EMA features. These plots are provided to visually demonstrate the model's performance on different individuals in the test dataset. The objective was to predict the step count for the following single timestep based on a sequence of step counts from the previous six days.

Two plots for participants 25 and 74 illustrate examples of poor model performance. The predicted step counts do not closely follow the actual values. The model often fails to capture the overall pattern of the step counts over time, missing several peaks where the actual steps increased sharply. At times, the predictions move in a different direction from the observed data. This shows that the model was unable to adequately learn the PA patterns for these participants, resulting in relatively large prediction errors of 18.75% for participant 25 and 50.0% for participant 74.

In contrast, the other plots (Participant 56 and 109) demonstrate good model performance. The predicted step counts closely followed the actual values, with the lines mostly overlapping. The model was able to capture temporal dependencies in step counts over time. These participants had some of the highest percentages of good predictions, exceeding 80%, which is reflected in the close alignment between the predicted and actual values.

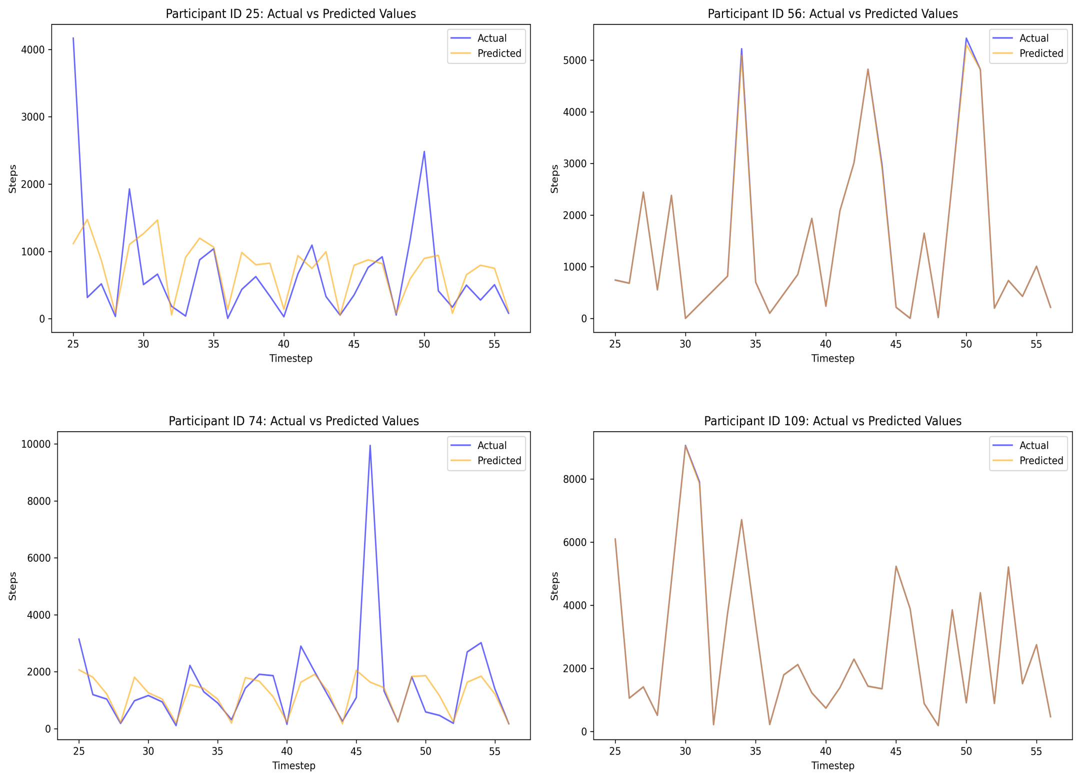

Figure 11: Prediction plots for selected participants. The blue line shows the actual step counts, while the orange line shows the model's predicted values.

In summary, the results of the longitudinal analysis showed that the most optimal input length for predicting PA for the next day (measured in four time steps) was seven days of step count data without EMA, using the LightGBM model (Steps only). Similarly, for predicting PA at a single timestep, the model performed best when using PA data from the previous six days with the LightGBM model (Steps only).

5 Discussion

The general aim of this thesis was to investigate different machine learning and deep learning methods and select the models that best predict PA, following two objectives. The first was to identify key predictors associated with PA and related outcomes of mild depression and risk of fall. The other objective aimed to find the optimal window size of the previous step counts needed to forecast PA correctly. First, the most important results of the two research questions will be discussed before addressing the limitations of the methods and ideas for future work.

5.1 Objective 1: The cross-sectional analysis

Among the different models evaluated for predicting the GDS category, the LightGBM model demonstrated the best overall performance. Within this model, the most important predictor was a specific item from the IPAQ. Indicating the strong association between specific self-reported PA behavior and mild depression status. The next important predictor was the quadriceps strength on the left side, measured in kilograms. Showing that lower limb strength was relevant for distinguishing between individuals with mild depression and those without depression. However, this needs to be interpreted with caution due to some limited performance metrics (e.g., F1 score) and sample size.

In the study by Song et al. [26], a sample of 7880 older adults in China was used to develop and evaluate a LightGBM model for predicting depression that was assessed using the CESD-10 scale. Their model achieved a Receiver Operating Characteristic Area Under the Curve (ROC AUC) value of 0.738. The most important predictors identified by the model included self-rated health and nighttime sleep duration, underscoring their significant roles in the occurrence of mild depression among older adults. These results differ from the predictive factors identified in this thesis. Nevertheless, because this thesis is based on a smaller sample size and shows different predictive factors compared to the much larger study by Song et al., and considering the limited performance of some of the metrics of LightGBM (F1 score of 0.571) presented at table 4, the identified predictive factors in the thesis for GDS may have limited reliability.

As for the risk of fall prediction model using LightGBM, which outperformed the other models, the most important feature was the quadriceps strength of the right leg. This shows the important role of lower limb muscle strength in maintaining balance and preventing falls among older adults. The other variables did not have a notable effect on the classification of this outcome due to a small importance score of less than 0.10. However, the reliability of these predictive variables is severely limited due to the low predictive performance of the model (PR AUC of 0.381).

In contrast to the results of the analysis of the thesis, Liang et al. [27] developed different machine learning classification models for falling, and they used posturographic data from 215 community-dwelling older adults. For classification based on fall history in the prior year, they employed ensemble classifiers, and the models achieved an ROC AUC of around 0.7.

Unlike Liang et al. [27], who found posturographic factors to be the most important predictors of risk of fall, the LightGBM model in this thesis did not find any balance control-related variables that were important predictors. This difference could be due to the smaller sample size of the cross-sectional data, which limited the ability to detect strong associations. Another possibility is that other factors in the cross-sectional data, such as the quadriceps strength of the right leg, may have a stronger influence on the risk of fall, making the effect of balance measures less influential. Further research with a larger number of participants and more specific balance tests may help to better understand these associations.

With regards to the IPAQ category, the LightGBM had superior overall performance compared to the other models. Exercise motivation had the most influence in classifying PA levels. The other factors were not as predictive (importance score was less than 0.10)

As for the IPAQ as a continuous measurement, the LightGBM model showed that the importance of physiological status and perceived quality of life in predicting PA as a continuous measurement in the cross-sectional analysis. But their importance scores were small (less than 0.10).

In general, the models identified certain variables as important predictors. However, their overall performance was generally limited. As a result, these findings are not very reliable and should be interpreted carefully, since the models might not have fully captured the true relationships between the predictors and the outcomes.

5.2 Objective 2: The longitudinal analysis

To address the second objective of the study, the LightGBM model using only lagged step counts was selected due to its consistently superior performance compared to other models. When forecasting PA for a full day, a sequence length of seven days (28 time steps) yielded the best results. The inclusion of psychological, contextual, and other EMA variables failed to enhance the prediction of next-day step counts, as model performance slightly deteriorated.

Similarly, when predicting the number of steps at a single future time point, using a six-day window provided the best performance. The inclusion of EMA features did not improve the prediction performance. Highlighting that recent step counts alone are more informative predictors of short-term physical activity.

Mamun et al. [28] conducted a study utilizing data collected from Fitbit Charge 2 wearable devices and smartphone applications BeWell24 and SleepWell24. The study included 99 participants, many of whom had more than 100 days of recorded observations. The authors employed LSTM models with a window size of seven days to predict the next day's physical activity of total step counts per day. They used multimodal features combining daily app engagement metrics, such as minutes used and times opened, along with physical activity measures, including sedentary duration, total device wear time, and other features. The final LSTM model achieved an MAE of 1677 steps for the prediabetic dataset and 2152 steps for the sleep dataset in forecasting the next day's step counts. In contrast to Mamun et al. [28], this thesis predicts physical activity using step counts divided into four three-hour time segments per day, rather than using total daily step counts. The final model developed here uses data from a seven-day window and relies only on step counts and time of day as input. This model achieved a MedAE of 414 steps (MedAE/Median of 0.31) in forecasting the next day's activity across four time segments.

With regard to the model combinations using fixed sequence lengths of six days for specific time segments, the analysis revealed notable differences in predictive performance dependent on the input-target temporal alignment. The LightGBM model using only lagged step counts achieved the best performance for within-segment predictions, specifically for afternoon-to-afternoon and evening-to-evening forecasts. Cross-segment configurations showed that forecasting morning targets was challenging, especially from noon, afternoon, or evening PA. In contrast, afternoon and evening targets were less difficult to forecast.

Adding EMA variables, such as contextual and psychological features, did not improve the performance of the models, including LightGBM, GRU, and LSTM, in most tasks, such as full day forecasting, single time step forecast, and different configurations of time segment inputs and targets.

The LOO analysis showed that for 43% of the participants, the LightGBM without EMA features model achieved a success rate of at least 80% when forecasting a single time step. However, for the remaining participants, the success rate was considerably lower. For these participants with lower performance, using only previous step counts or including EMA inputs did not help the model to learn their PA patterns accurately. These differences may reflect greater variability or irregularity in the daily activity patterns, which may limit the model's ability to learn the PA patterns of these participants.

An additional analysis was performed to determine if participants who met the success criterion of having correct predictions differed from those who did not based on demographic or clinical variables such as age, gender, fall risk, IPAQ category, and mild depression status. The results showed no statistically significant differences, indicating that variations in predictive performance were not systematically linked to these factors. This can be due to other unmeasured factors that may be influencing the differences in model performance across different participants.

5.3 Limitations and drawbacks of the methods

In both the cross-sectional and longitudinal analyses, several candidate models were trained, and the model with the best performance according to the selected evaluation metric was chosen. This approach can have some limitations. Different models may capture different patterns in the data. By selecting only one model, these additional patterns were omitted, and possible improvements from combining different model predictions, such as through stacking methods, were not considered [29].

The performance of the model for predicting the GDS category was relatively poor for some metrics. This may be due to the limited sample size or the small number of participants in the study. Additionally, important factors such as additional sleep patterns were not included in the cross-sectional dataset, which could have affected the model's ability to correctly predict depression status [26].

The drawback of the risk of fall prediction model included low performance caused by class imbalance and a small dataset size. These factors limited the model's ability to detect strong associations compared to other studies [27].

As for the limitation of the longitudinal prediction modeling, the final selected LightGBM model without EMA achieved accurate predictions for some participants, but lower performance for others. One possibility is that for some participants, relying solely on previous step counts or adding features from EMA did not provide useful information for predicting their PA, which may indicate that their activity patterns were influenced by external, unmeasured factors such as environmental conditions or other variables that were not measured in the longitudinal dataset. Another possibility is that some participants shared similar physical activity patterns, allowing the model to learn these patterns from certain individuals and generalize them to others with similar PA behaviors.

Moreover, another drawback is that LightGBM, being a model primarily developed for tabular data, may not be ideally suited to capture temporal dependencies in time series data. Unlike RNNs or other methods specifically developed to learn complex temporal patterns for forecasting, LightGBM might have limitations in effectively modeling the sequential nature of physical activity data and may not learn more temporal PA patterns that are present in the dataset without comprehensive feature engineering [24]. Therefore, while the results of the final model provide valuable insights, they should be interpreted with caution, given these potential limitations in capturing temporal dynamics.

In addition, hyperparameter tuning using Bayesian optimization was conducted on the final selected LightGBM (Steps only) model. However, this tuning process did not result in improved parameter values compared to those obtained before the optimization. This is due to the number of parameters to tune (nine), combined with a limited number of iterations, which restricted the optimization from finding better parameter combinations. For the GRU and LSTM models, no formal hyperparameter tuning was performed; several different choices of model structures were tested initially, and the best-performing setup was chosen and used consistently across all related models.

Furthermore, the modeling involved transforming the outcome variable of step count using the Yeo-Johnson transformation, training the models using these transformed values, and then back-transforming the predictions for evaluation. However, back-transformation can introduce bias into the predicted values [30].

5.4 Ideas for future work and research

Future work should include collecting more data (increasing the number of participants and other types of data that could influence the level of physical activity, such as weather, sleep, or air quality) for both the cross-sectional and longitudinal datasets. Having larger and more diverse data can help improve the reliability of the predictive models and allow for a better understanding of which variables serve as reliable predictors. This increased data availability may also support capturing a wider range of PA patterns and behaviors, helping the models to generalize better across different participants.

Regarding the methodology, future work should explore a broader range of modeling techniques. Specifically, additional deep learning methods such as Temporal Convolutional Neural Networks (TCNs) could be investigated alongside the recurrent models already used for the longitudinal analysis. Combining these approaches with formal hyperparameter tuning methods, like Bayesian optimization, for all models could further improve predictive performance. This would allow for a more thorough comparison of different algorithms and help identify the most effective modeling strategies for forecasting PA [24].

Additionally, stacking methods should be investigated for both cross-sectional and longitudinal data analysis. Stacking is an ensemble learning method where predictions from multiple base models at the first level are used as input features for a meta-model at the next level. The meta-model combines the predictions from the base models. It takes into account differences caused by various parameter settings and different subsets of data used to train the base models. This approach can improve prediction performance by combining the strengths of the base models and reducing their overall errors. Examples of this improvement have been shown in time series forecasting and logistic regression with imbalanced data [29].

Furthermore, future research should investigate bias correction techniques for the back-transformation of predicted values or explore alternative methods to handle the skewness of step count data in longitudinal models [30].

6 Conclusion

This thesis examined the application of machine learning and deep learning techniques to predict physical activity levels in older adults, using both cross-sectional and longitudinal datasets. Several types of models were evaluated, including Linear and Logistic Regression, LightGBM, RNN such as GRU and LSTM, and Elastic Net.

In the cross-sectional analysis, models were developed to predict PA levels and related outcomes of falling risk and mild depression status. The LightGBM model achieved the best overall results for this task. The most important predictor identified for the IPAQ category outcome was an item from the ESES. Showing that particular aspects of exercise self-efficacy were important in differentiating between high and moderate physical activity levels.

In the longitudinal analysis, time series models were trained to predict step counts using sequences of past observations. The results showed that a seven-day input sequence provided the best predictive performance for full-day PA, measured in four time steps. Six-day input was the optimal window for single-time-step forecasts. However, model performance varied across individuals, and the models had limited ability to generalize correctly across all participants.

Overall, this thesis demonstrates the potential of combining wearable sensor data and machine learning methods to understand and predict physical activity in older adults. Some predictive models performed well, particularly for participants whose physical activity could be accurately predicted from their previous observations. However, further work is necessary to improve the generalizability of these models and to facilitate personalized health interventions.

References

[1] World Health Organization. *Ageing and health.* 2024. URL: `https://www.who.int/news-room/fact-sheets/detail/ageing-and-health`.

[2] Thomas Vogel et al. “Health benefits of physical activity in older patients: a review”. In: *International Journal of Clinical Practice* (2009).

[3] Birgitta Langhammer, Astrid Bergland, and Elisabeth Rydwik. “The Importance of Physical Activity Exercise among Older People”. In: *BioMed Research International* (2018).

[4] Marina B. Pinheiro et al. “Impact of physical activity programs and services for older adults: a rapid review”. In: *International Journal of Behavioral Nutrition and Physical Activity* (2022).

[5] Denise Taylor. “Physical activity is medicine for older adults”. In: *Postgraduate Medical Journal* (2014).

[6] Kyungmi Lee et al. “Using digital phenotyping to understand health-related outcomes: A scoping review”. In: *International Journal of Medical Informatics* (2023).

[7] Kim Daniels et al. “From Steps to Context: Optimizing Digital Phenotyping for Physical Activity Monitoring in Older Adults by Integrating Wearable Data and Ecological Momentary Assessment”. In: *Sensors* (2025).

[8] Yifan Lu et al. “Association Between Physical Activity and Risk of Depression: A Systematic Review and Meta-Analysis of Prospective Studies”. In: *International Journal of Environmental Research and Public Health* (2022).

[9] Yingbo Zhang et al. “The comprehensive clinical benefits of digital phenotyping: from broad adoption to full impact”. In: *npj Digital Medicine* (2025).

[10] Ezgi Hasret Kozan Cikirikci and Melek Nihal Esin. “The impact of machine learning on physical activity–related health outcomes: A systematic review and meta-analysis”. In: *International Nursing Review* (2025).

[11] Mo Zhou et al. “Evaluating Machine Learning–Based Automated Personalized Daily Step Goals Delivered Through a Mobile Phone App: Randomized Controlled Trial”. In: *JMIR mHealth and uHealth* (2018).

[12] Schenelle Dayna Dlima et al. “Digital Phenotyping in Health Using Machine Learning Approaches: Scoping Review”. In: *JMIR Bioinformatics and Biotechnology* (2022).

[13] Kim Daniels et al. “Characterising physical activity patterns in community-dwelling older adults using digital phenotyping: a 2-week observational study protocol”. In: *BMJ Open* (2025).

[14] Max Kuhn. “Building Predictive Models in R Using the caret Package”. In: *Journal of Statistical Software* (2008).

[15] Trevor Hastie and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.

[16] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. “Regularization Paths for Generalized Linear Models via Coordinate Descent”. In: *Journal of Statistical Software* (2010).

[17] Guolin Ke et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS).* 2017.

[18] Michael W. Browne. “Cross-validation methods”. In: *Journal of Mathematical Psychology* (2000).

[19] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems.* 2012.

[20] Alexei Botchkarev. “Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology”. In: (2024).

[21] Margherita Grandini, Enrico Bagli, and Giorgio Visani. *Metrics for Multi-Class Classification: An Overview.* Tech. rep. CRIF S.p.A. and Department of Computer Science, University of Bologna, 2020.

[22] Takaya Saito and Marc Rehmsmeier. “The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets”. In: *PLOS ONE* (2015).

[23] Filippo Maria Bianchi et al. “An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting”. In: *arXiv preprint arXiv:1705.04378* (2018).

[24] Bryan Lim and Stefan Zohren. “Time-series forecasting with deep learning: a survey”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* (2021).

[25] Sanford Weisberg. *Yeo-Johnson Power Transformations.* Tech. rep. Department of Applied Statistics, University of Minnesota, 2001.

[26] Yan Li Qing Song et al. “Machine Learning Algorithms to Predict Depression in Older Adults in China: A Cross-Sectional Study”. In: *Frontiers in Public Health* (2025).

[27] Huey-Wen Liang et al. “Fall risk classification with posturographic parameters in community-dwelling older adults: a machine learning and explainable artificial intelligence approach”. In: *Journal of NeuroEngineering and Rehabilitation* (2024).

[28] Abdullah Mamun et al. “Multimodal Physical Activity Forecasting in Free-Living Clinical Settings: Hunting Opportunities for Just-in-Time Interventions”. In: *arXiv preprint arXiv:2410.09643* (2024).

[29] Bohdan Pavlyshenko. “Using Stacking Approaches for Machine Learning Models”. In: *Proceedings of the IEEE Second International Conference on Data Stream Mining & Processing (DSMP).* 2018.

[30] Sushant More. “Identifying and Overcoming Transformation Bias in Forecasting Models”. In: *arXiv preprint arXiv:2208.12264* (2022).

7 Software code

The full code is available at this GitHub Repository
`https://github.com/AnasNazar98/Thesis_software_code.git`

The software codes of a few selected models are presented in this document; the complete software files and code are in the repository.

Cross-sectional R code

```r
# imputing the cross-sectional data

rm(list = ls())
library(tidyverse)
library(skimr)
library(magrittr)
library(readxl)
library(writexl)

###############################################################################
# Cross-sectional data
###############################################################################


cross <- read_excel('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    Clinical_Anas.xlsx')

str(cross)
glimpse(cross)

cross <- cross %>%
  mutate(across(starts_with('ipaq_'), ~ ifelse(. == 'NULL', NA, .)))

cross <- cross %>%
  mutate(across(starts_with('ipaq_'), ~ ifelse(. == 'ik heb geen matige
      lichamelijke activiteiten gedaan', 0, .)))

cross <- cross %>%
  mutate(across(starts_with('borg'), ~ ifelse(. == 'NULL', NA, .)))

cross <- cross %>%
  mutate(across(where(is.character), ~ na_if(., 'NULL')))

cross <- cross %>%
  mutate(across(everything(), ~ ifelse(. == 'Ja', 1, .)))

cross <- cross %>%
  mutate(across(everything(), ~ ifelse(. == 'Universitair onderwijs', NA, .)))

cross$gds_category <- ifelse(cross$gds_category == 'Mild depressed', 1, 0)

cross <- cross %>%
  mutate(IPAQ_category = case_when(
    IPAQ_category == 'Low' ~ 1,
    IPAQ_category == 'moderate' ~ 2,
    IPAQ_category == 'high' ~ 3,

  ))

cross <- cross %>%
  mutate(across(where(is.character), as.numeric))

# processed data for modelling
write_xlsx(cross, 'C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/cross_new.
    xlsx')

library(mice)

cross <- cross %>%
  mutate(
    diploma = as.factor(diploma),
    kinvent_hand_l = as.numeric(kinvent_hand_l),
    IPAQ_category = as.ordered(IPAQ_category)
  )

imputation_methods <- make.method(cross)

imputation_methods['diploma'] <- 'polr'
imputation_methods['bloodpressure_sys'] <- 'pmm'
imputation_methods['bloodpressure_dia'] <- 'pmm'
imputation_methods['heartrate'] <- 'pmm'
imputation_methods['saturation_mea_post'] <- 'pmm'
imputation_methods['heartbeat_post'] <- 'pmm'
imputation_methods['kinvent_hand_l'] <- 'logreg'
imputation_methods['score_hand_l'] <- 'pmm'
imputation_methods['score_hand_r'] <- 'pmm'
imputation_methods['score_qua_left'] <- 'pmm'
imputation_methods['score_qua_right'] <- 'pmm'
imputation_methods['sit_reach_values_1'] <- 'pmm'
imputation_methods['sit_reach_values_2'] <- 'pmm'
imputation_methods['sit_reach_values_3'] <- 'pmm'
imputation_methods['sit_reach_highest'] <- 'pmm'
imputation_methods['symmetry'] <- 'pmm'
imputation_methods['cadence'] <- 'pmm'
imputation_methods['speed'] <- 'pmm'
imputation_methods['stance_time_left'] <- 'pmm'
imputation_methods['stance_time_right'] <- 'pmm'
imputation_methods['swing_time_left'] <- 'pmm'
imputation_methods['swing_time_right'] <- 'pmm'
imputation_methods['double_support'] <- 'pmm'
imputation_methods['propulsion_dur_left'] <- 'pmm'
imputation_methods['propulsion_dur_right'] <- 'pmm'
imputation_methods['flatfoot_left'] <- 'pmm'
imputation_methods['flatfoot_right'] <- 'pmm'
imputation_methods['loading_left'] <- 'pmm'
imputation_methods['loading_right'] <- 'pmm'
imputation_methods['propulsion_ratio_left'] <- 'pmm'
imputation_methods['propulsion_ratio_righ'] <- 'pmm'
imputation_methods['pro_sup_angle_heelgr_l'] <- 'pmm'
imputation_methods['pro_sup_angle_flat_l'] <- 'pmm'
imputation_methods['pro_sup_angle_heelli_l'] <- 'pmm'
imputation_methods['pro_sup_angle_toeli_l'] <- 'pmm'
```

```r
imputation_methods['pro_sup_angle_heelgr_r'] <- 'pmm'
imputation_methods['pro_sup_angle_flat_r'] <- 'pmm'
imputation_methods['pro_sup_angle_heelli_r'] <- 'pmm'
imputation_methods['pro_sup_angle_toeli_r'] <- 'pmm'
imputation_methods['step_progr_angle_left'] <- 'pmm'
imputation_methods['step_progr_angle_right'] <- 'pmm'
imputation_methods['circumduction_left'] <- 'pmm'
imputation_methods['circumduction_right'] <- 'pmm'
imputation_methods['clearance_left'] <- 'pmm'
imputation_methods['clearance_right'] <- 'pmm'
imputation_methods['steppage_heel_left'] <- 'pmm'
imputation_methods['steppage_heel_right'] <- 'pmm'
imputation_methods['steppage_toe_left'] <- 'pmm'
imputation_methods['steppage_toe_right'] <- 'pmm'

library(doParallel)
library(finetune)

# processing
ncores <- parallel::detectCores() - 3
cl <- makePSOCKcluster(ncores)
registerDoParallel(cl)

imputed_data <- mice(cross, method = imputation_methods, m = 10, maxit = 10)

cross_imputed <- complete(imputed_data, 10)
view(cross_imputed)
# saving the imputed data for modelling
write_xlsx(cross_imputed, 'C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    all_imputations.xlsx')
##################################################


rm(list = ls())
library(tidyverse)
library(dplyr)
library(ggplot2)
library(skimr)
library(magrittr)
library(readxl)
library(writexl)
library(corrplot)
library(glmnet)
library(caret)
library(pROC)
library(xgboost)
library(PRROC)
library(tidymodels)
library(vip)
library(dials)
library(purrr)
library(tibble)
library(yardstick)
library(recipes)
library(finetune)
library(future)
```

```r

###################################################################################
# Logistic Regression IPAQ category
###################################################################################



rm(list = ls())
seed <- 42

sheet_names <- excel_sheets("C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    imputations/all_imputations.xlsx")


for (i in seq_along(sheet_names)){
  sheet_data <- read_excel("C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
      imputations/all_imputations.xlsx",
                           sheet = sheet_names[i])
  assign(paste0("cross", i), sheet_data, envir = .GlobalEnv)
}
cross_all <- list(cross1, cross2, cross3, cross4, cross5,
                  cross6, cross7, cross8, cross9, cross10)

gender <- read_xlsx('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    Qualtrics_vragenlijst_fysiek_final_241024.xlsx')


data_train <- list()
data_test <- list()

coef_df_list <- list()

predictions_list <- list()

length <- 1
for (i in 1:length){
cross <- cross_all[[i]]

    cross <- cross_all[[i]]

    cross$gender <- gender$gender


    cross <- cross %>%
      filter(!IPAQ_category == "1") %>%
      mutate(IPAQ_category = ifelse(IPAQ_category == "2", 0, 1))


    outcome <- factor(ifelse(cross$IPAQ_category == '1', 'Yes', 'No'), levels =
        c('Yes', 'No'))


    cross <- cross %>%
      mutate(across(everything(), ~ as.numeric(as.character(.))))




    for (col in names(cross)) {
      unique_vals <- length(unique(na.omit(cross[[col]])))
      if (unique_vals <= 5) {
        cross[[col]] <- as.factor(cross[[col]])
      }
    }


    cross <- cross %>%
      mutate(across(
        where(is.factor),
        ~ if (all(levels(.) %in% c("1", "2"))) {
          factor(ifelse(. == "2", "0", "1"), levels = c("0", "1"))
        } else {
          .
        }
      ))






    cross <- cross %>%
      dplyr::select(-participant_id, -starts_with("ipaq"), -starts_with('IPAQ')
        )

    cross$IPAQ_category <- outcome

    cross <- cross %>% mutate(case_wts = ifelse(IPAQ_category == "Yes", 1, 5),
                              case_wts = importance_weights(case_wts))

    model <- 'Logistic Regression'
    label <- 'IPAQ Category'






    cross$IPAQ_category <- outcome

    set.seed(seed)
    data_split <- initial_split(cross, strata = IPAQ_category, prop = 0.70)
    data_train[[i]] <- training(data_split)
    data_test[[i]] <- testing(data_split)



    spec_default <- logistic_reg() %>%
      set_engine("glm") %>%
      set_mode("classification")


    rec_default <- recipe(IPAQ_category ~ ., data = data_train[[i]]) %>%
      step_unknown(all_nominal_predictors(), new_level = "unknown") %>%
      step_dummy(all_nominal_predictors()) %>%
      step_zv(all_predictors()) %>%
      step_normalize(all_numeric_predictors()) %>%
      step_corr(all_numeric_predictors(), threshold = 0.6)


    wf_default <- workflow() %>%
      add_recipe(rec_default) %>%
      add_model(spec_default) %>% add_case_weights(case_wts)



    library(FSelectorRcpp)


    rec_baked <- prep(rec_default, training = data_train[[i]])

    data_train_for_vip <- bake(rec_baked, new_data = data_train[[i]])

    data_train_for_vip <- data_train_for_vip %>% dplyr::select(
      -case_wts)



    vi_df <- information_gain(IPAQ_category ~ . - case_wts, data = data_train[[
       i]])

    top_vars <- vi_df %>%
      arrange(desc(importance)) %>%
      slice_head(n = 80) %>%
      pull(attributes)

    library(stringr)

    cleaned_vars <- top_vars %>%
      str_remove("_X\\d+$") %>%
      unique()




    data_train[[i]] <- data_train[[i]] %>% dplyr::select(all_of(c(cleaned_vars,
        "IPAQ_category", "case_wts")))
    data_test[[i]]  <- data_test[[i]] %>% dplyr::select(all_of(c(cleaned_vars,
       "IPAQ_category")))
    data_test[[i]]  <- data_test[[i]] %>% dplyr::select(all_of(c(cleaned_vars,
       "IPAQ_category")))


    rec_default <- recipe(IPAQ_category ~ ., data = data_train[[i]]) %>%
      step_unknown(all_nominal_predictors(), new_level = "unknown") %>%
      step_dummy(all_nominal_predictors()) %>%
      step_zv(all_predictors()) %>%
      step_normalize(all_numeric_predictors()) %>%
      step_corr(all_numeric_predictors(), threshold = 0.6)



    wf_default <- workflow() %>%
      add_recipe(rec_default) %>%
      add_model(spec_default) %>% add_case_weights(case_wts)




    default_res <- last_fit(
      wf_default,
      split = data_split,
      metrics = metric_set(
        yardstick::f_meas,
        yardstick::precision,
        yardstick::recall,
        yardstick::spec,
        yardstick::accuracy,
        yardstick::bal_accuracy

        , yardstick::pr_auc

      )
    )


    collect_metrics(default_res)

    preds <- collect_predictions(default_res) %>%
      mutate(.pred_class = factor(if_else(.pred_Yes >= 0.5, "Yes", "No"),
          levels = c("Yes", "No")))

    collect_metrics(default_res)
    conf_mat(preds, truth = IPAQ_category, estimate = .pred_class)



    final_model <- extract_fit_parsnip(default_res$.workflow[[1]])
    summary(final_model$fit)



    coef_df <- coef(summary(final_model$fit)) %>%
      as.data.frame() %>%
      rownames_to_column("feature") %>%
      dplyr::select(feature, coefficient = Estimate)

    coef_df_list[[i]] <- coef_df




    test_probs <- preds$.pred_Yes
```

```
    test_preds <- preds$.pred_class
    truth <- data_test[[i]]$IPAQ_category


    predictions_list[[i]] <- tibble(
      truth = truth,
      .pred_class = test_preds,
      .pred_Yes = test_probs
    )

}







  combined_coefs <- bind_rows(coef_df_list, .id = "imputation")
  combined_predictions <- bind_rows(predictions_list, .id = "imputation")





  all_preds <- bind_rows(predictions_list, .id = "imputation")


  pred_list <- list()

  for (i in 1:length) {
    pred_list[[i]] <- predictions_list[[i]]$.pred_Yes
  }

  avg_preds <- rowMeans(do.call(cbind, pred_list))

  truth <- predictions_list[[1]]$truth

  final_avg_preds <- data.frame(
    .pred_Yes = avg_preds,
    truth = factor(truth, levels = c("Yes", "No")),
    .pred_class = factor(ifelse(avg_preds >= 0.5, "Yes", "No"), levels = c("Yes
       ", "No"))
  )
  conf_mat(final_avg_preds, truth = truth, estimate = .pred_class)



  truth <- final_avg_preds$truth
  pred <- final_avg_preds$.pred_class
  probs <- final_avg_preds$.pred_Yes

  truth <- factor(truth, levels = c("Yes", "No"))
  pred <- factor(pred, levels = c("Yes", "No"))

  f1          <- f_meas_vec(truth, pred)
```

```
precision    <- precision_vec(truth, pred)
recall       <- recall_vec(truth, pred)
specificity  <- specificity_vec(truth, pred)
accuracy     <- accuracy_vec(truth, pred)
bal_accuracy <- bal_accuracy_vec(truth, pred)
pr_auc       <- pr_auc_vec(truth, probs, event_level = "first")


metrics <- tibble(
  Metric = c(
    "F1 Score",
    "Precision",
    "Recall (Sensitivity)",
    "Specificity",
    "Accuracy",
    "Bal. Accuracy",
    "PR_AUC"
  ),
  Value = c(
    f1,
    precision,
    recall,
    specificity,
    accuracy,
    bal_accuracy,
    pr_auc
  )
)
(metrics)
conf_mat(final_avg_preds, truth = truth, estimate = .pred_class)


model <- 'Logistic regression'
label <- 'IPAQ Category'

all_coefs <- bind_rows(coef_df_list, .id = "imputation")

pooled_coefs <- all_coefs %>%
  group_by(feature) %>%
  summarise(mean_coef = mean(coefficient, na.rm = TRUE)) %>%
  ungroup()
pooled_coefs <- pooled_coefs %>%
  rename(coef = mean_coef) %>%
  filter(coef != 0)

intercept <- pooled_coefs %>%
  filter(feature == "(Intercept)") %>%
  pull(coef)

coefs <- pooled_coefs %>%
  filter(feature != "(Intercept)")



coef_df <- pooled_coefs %>%
  filter(feature != "(Intercept)", coef != 0) %>%
```

```r
    mutate(
      direction = ifelse(coef > 0, "Positive", "Negative"),
      abs_coef = abs(coef)
    ) %>%
    slice_max(order_by = abs_coef, n = 10)



  model <- 'Logistic Regression'
  label <- 'IPAQ Category'

  ggplot(coef_df, aes(x = reorder(feature, abs_coef), y = abs_coef, fill =
      direction)) +
    geom_col() +
    coord_flip() +
    scale_fill_manual(values = c("Positive" = "dodgerblue", "Negative" = "red")
      ) +
    labs(
      title = paste('Most predictive features for\n', label, 'using', model),
      x = "Feature",
      y = "Importance (|Coefficient|)",
      fill = "Effect Direction"
    ) +
    theme_minimal()

################################################################################

################################################################################

################################################################################

################################################################################

# Elastic Net IPAQ category
################################################################################


rm(list = ls())
seed <- 42

sheet_names <- excel_sheets("C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    imputations/all_imputations.xlsx")

for (i in seq_along(sheet_names)){
  sheet_data <- read_excel("C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
      imputations/all_imputations.xlsx",
                           sheet = sheet_names[i])
  assign(paste0("cross", i), sheet_data, envir = .GlobalEnv)
}
cross_all <- list(cross1, cross2, cross3, cross4, cross5,
                  cross6, cross7, cross8, cross9, cross10)

gender <- read_xlsx('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    Qualtrics_vragenlijst_fysiek_final_241024.xlsx')


```

```r
data_train <- list()
data_test <- list()

coef_df_list <- list()

predictions_list <- list()


for (i in 1:10) {
  cross <- cross_all[[i]]


  cross$gender <- gender$gender

  cross <- cross %>%
    filter(!IPAQ_category == "1") %>%
    mutate(IPAQ_category = ifelse(IPAQ_category == "2", 0, 1))

  cross$sit_reach_values_3[is.na(cross$sit_reach_values_3)] <- 0

  outcome <- factor(ifelse(cross$IPAQ_category == '1', 'Yes', 'No'), levels = c
      ('Yes', 'No'))

  cross <- cross %>%
    mutate(across(everything(), ~ as.numeric(as.character(.))))

  zero_var_indices <- nearZeroVar(cross)

  cross <- cross[, -zero_var_indices]


for (col in names(cross)) {
  unique_vals <- length(unique(na.omit(cross[[col]])))
  if (unique_vals <= 5) {
    cross[[col]] <- as.factor(cross[[col]])
  }
}


cross <- cross %>%
  mutate(across(
    where(is.factor),
    ~ if (all(levels(.) %in% c("1", "2"))) {
      factor(ifelse(. == "2", "0", "1"), levels = c("0", "1"))
    } else {
      .
    }
  ))



outcome <- factor(ifelse(cross$IPAQ_category == '1', 'Yes', 'No'), levels = c('
    Yes', 'No'))



```

```
cross <- cross %>%
  dplyr::select(-participant_id, -starts_with("ipaq"), -starts_with("IPAQ"))

cross$IPAQ_category <- outcome


cross <- cross %>% mutate(case_wts = ifelse(IPAQ_category == "Yes", 1, 2.5),
                          case_wts = importance_weights(case_wts))

model <- 'Elastic Net'
label <- 'IPAQ category'

set.seed(seed)
data_split <- initial_split(cross, strata = IPAQ_category, prop = 0.70)
data_train[[i]] <- training(data_split)
data_test[[i]] <- testing(data_split)
}

table(cross$IPAQ_category)
(start_time <- Sys.time())
for(i in 1:10){
set.seed(seed)
data_folds <- vfold_cv(data_train[[i]], strata = IPAQ_category, v = nrow(
    data_train[[i]]))
data_folds <- vfold_cv(data_train[[i]], strata = IPAQ_category, v = 10
)

library(tune)
library(doParallel)

spec <- logistic_reg(
  penalty = tune()
  ,mixture = tune()
) %>%
  set_engine("glmnet"
  ) %>%
  set_mode("classification")

params <- parameters(
  penalty(range = c(-5, 1))
  ,mixture(range = c(0, 1)))



rec <- recipe(IPAQ_category ~ ., data = data_train[[i]]) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())




wf <- workflow() %>%
  add_recipe(rec) %>%
  add_model(spec) %>% add_case_weights(case_wts)


```

```
rec_prep <- prep(rec, training = data_train[[i]])
processed_data <- bake(rec_prep, new_data = NULL)




plan(sequential)
plan(multisession, workers = parallel::detectCores() - 2, gc = TRUE)

set.seed(seed)
res <- tune_bayes(
  wf,
  resamples = data_folds,
  param_info = params,
  initial = 20,
  iter = 20,
  metrics = metric_set(
    f_meas,
    yardstick::precision,

  )
  ,control = control_bayes(
    verbose = T,
    no_improve = 20,
    seed = 123,
    save_pred = TRUE,
    allow_par = TRUE
  )
)

plan(sequential)
plan()

ipaq_cat_en_res <- res



best_parms <- select_best(res, metric = "precision")

set.seed(seed)
final <- finalize_workflow(wf, best_parms)

final_res <- last_fit(final, data_split, metrics = metric_set(
  f_meas,
  yardstick::precision,
  yardstick::recall,
  yardstick::specificity,
  yardstick::accuracy,
  yardstick::bal_accuracy,
  pr_auc

))
collect_metrics(final_res)

final_fit <- fit(final, data = data_train[[i]])

```

```
(glmnet_model <- extract_fit_parsnip(final_fit)$fit)

(best_params <- select_best(res, metric = "precision"))
(best_lambda <- best_params$penalty)
(best_alpha <- best_params$mixture)

coefs <- coef(glmnet_model, s = best_lambda)

coef_df <- data.frame(
  feature = rownames(coefs),
  coefficient = as.vector((coefs)))

coef_df_list[[i]] <- coef_df

predictions_list[[i]] <- collect_predictions(final_res)
}
end_time <- Sys.time()
(parallel_time <- end_time - start_time)

library(writexl)



combined_coefs <- bind_rows(coef_df_list, .id = "imputation")
combined_predictions <- bind_rows(predictions_list, .id = "imputation")





all_preds <- bind_rows(predictions_list, .id = "imputation")


pred_list <- list()

for (i in 1:10) {
pred_list[[i]] <- predictions_list[[i]]$.pred_Yes
}

avg_preds <- rowMeans(do.call(cbind, pred_list))

truth <- predictions_list[[1]]$IPAQ_category

final_avg_preds <- data.frame(
  .pred_Yes = avg_preds,
  truth = factor(truth, levels = c("Yes", "No")),
  .pred_class = factor(ifelse(avg_preds >= 0.5, "Yes", "No"), levels = c("Yes",
      "No"))
)


conf_mat(final_avg_preds, truth = truth, estimate = .pred_class)



truth <- final_avg_preds$truth
```

```r
pred <- final_avg_preds$.pred_class
probs <- final_avg_preds$.pred_Yes

truth <- factor(truth, levels = c("Yes", "No"))
pred <- factor(pred, levels = c("Yes", "No"))

f1           <- f_meas_vec(truth, pred)
precision    <- precision_vec(truth, pred)
recall       <- recall_vec(truth, pred)
specificity  <- specificity_vec(truth, pred)
accuracy     <- accuracy_vec(truth, pred)
bal_accuracy <- bal_accuracy_vec(truth, pred)
pr_auc       <- pr_auc_vec(truth, probs, event_level = "first")


metrics <- tibble(
  Metric = c(
    "F1 Score",
    "Precision",
    "Recall (Sensitivity)",
    "Specificity",
    "Accuracy",
    "Bal. Accuracy",
    "PR_AUC"
  ),
  Value = c(
    f1,
    precision,
    recall,
    specificity,
    accuracy,
    bal_accuracy,
    pr_auc
  )
)

print(metrics)
conf_mat(final_avg_preds, truth = truth, estimate = .pred_class)


model <- 'Elastic Net'
label <- 'GDS category'

all_coefs <- bind_rows(coef_df_list, .id = "imputation")

pooled_coefs <- all_coefs %>%
  group_by(feature) %>%
  summarise(mean_coef = mean(coefficient, na.rm = TRUE)) %>%
  ungroup()

pooled_coefs <- pooled_coefs %>%
  rename(coef = mean_coef) %>%
  filter(coef != 0)

intercept <- pooled_coefs %>%
  filter(feature == "(Intercept)") %>%
  pull(coef)

coefs <- pooled_coefs %>%
  filter(feature != "(Intercept)")



coef_df <- pooled_coefs %>%
  filter(feature != "(Intercept)", coef != 0) %>%
  mutate(
    direction = ifelse(coef > 0, "Positive", "Negative"),
    abs_coef = abs(coef)
  ) %>%
  slice_max(order_by = abs_coef , n = 10)



model <- 'Elastic Net'
label <- 'IPAQ category'

ggplot(coef_df, aes(x = reorder(feature, abs_coef), y = abs_coef, fill =
    direction)) +
  geom_col() +
  coord_flip() +
  scale_fill_manual(values = c("Positive" = "dodgerblue", "Negative" = "red"))
     +
  labs(
    title = paste('Most predictive features for\n', label, 'using', model),
    x = "Feature",
    y = "Importance (|Coefficient|)",
    fill = "Effect Direction"
  ) +
  theme_minimal()

################################################################################
################################################################################
################################################################################
################################################################################
# LightGBM ipaq category
################################################################################
rm(list = ls())
seed <- 42

cross <- read_excel('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    cross_processed.xlsx')

gender <- read_xlsx('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    Qualtrics_vragenlijst_fysiek_final_241024.xlsx')

cross$gender <- gender$gender

```

```r




cross <- cross %>%
  filter(!IPAQ_category == "1") %>%
  mutate(IPAQ_category = ifelse(IPAQ_category == "2", 0, 1))


outcome <- factor(ifelse(cross$IPAQ_category == '1', 'Yes', 'No'), levels = c('
    Yes', 'No'))

cross <- cross %>%
  mutate(across(everything(), ~ as.numeric(as.character(.))))

zero_var_indices <- nearZeroVar(cross)

cross <- cross[, -zero_var_indices]


for (col in names(cross)) {
  unique_vals <- length(unique(na.omit(cross[[col]])))
  if (unique_vals <= 5) {
    cross[[col]] <- as.factor(cross[[col]])
  }
}


cross <- cross %>%
  mutate(across(
    where(is.factor),
    ~ if (all(levels(.) %in% c("1", "2"))) {
      factor(ifelse(. == "2", "0", "1"), levels = c("0", "1"))
    } else {
      .
    }
  ))



outcome <- factor(ifelse(cross$IPAQ_category == '1', 'Yes', 'No'), levels = c('
    Yes', 'No'))




cross <- cross %>%
  dplyr::select(-participant_id, -starts_with("ipaq"), -starts_with("IPAQ"))

cross$IPAQ_category <- outcome


cross <- cross %>% mutate(case_wts = ifelse(IPAQ_category == "Yes", 1, 2),
                          case_wts = importance_weights(case_wts))


```

```
model <- 'Elastic Net'
label <- 'IPAQ category'


set.seed(seed)
data_split <- initial_split(cross, strata = IPAQ_category, prop = 0.7)
data_train <- training(data_split)
data_test <- testing(data_split)
library(bonsai)


spec_default <- boost_tree() %>%
  set_engine("lightgbm") %>%
  set_mode("classification")


rec_default <- recipe(IPAQ_category ~ ., data = data_train) %>%
  step_unknown(all_nominal_predictors(), new_level = "unknown") %>%

  step_dummy(all_nominal_predictors())

wf_default <- workflow() %>%
  add_recipe(rec_default) %>%
  add_model(spec_default) %>% add_case_weights(case_wts)










default_res <- last_fit(
  wf_default,
  split = data_split,
  metrics = metric_set(
    yardstick::f_meas,
    yardstick::precision,
    yardstick::recall,
    yardstick::spec,
    yardstick::accuracy,
    yardstick::bal_accuracy,
    yardstick::pr_auc
  )
)


collect_metrics(default_res)



preds <- collect_predictions(default_res) %>%
  mutate(.pred_class = factor(if_else(.pred_Yes >= 0.5, "Yes", "No"), levels =
      c("Yes", "No")))

```

```r

collect_metrics(default_res)

conf_mat(preds, truth = IPAQ_category, estimate = .pred_class)




fitted_model <- extract_fit_parsnip(default_res)

vip(fitted_model$fit, num_features = 10) +
  ggtitle(paste('Most predictive features for\n', label, 'using', model))



set.seed(seed)
spec <- boost_tree(
  trees = tune(),
  tree_depth = tune(),
  min_n = tune(),
  loss_reduction = tune(),
  sample_size = tune(),
  learn_rate = tune()
) %>%
  set_engine("lightgbm",
             lambda_l1 = tune(),
             lambda_l2 = tune()
             , num_leaves = tune()) %>%
  set_mode("classification")


library(dials)
set.seed(seed)
params <- parameters(
  trees(),
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  learn_rate(),

  lambda_l1 = penalty(range = c(-5, 1)),
  lambda_l2 = penalty(range = c(-5, 1))
  , num_leaves()
)


rec <- recipe(IPAQ_category ~ ., data = data_train) %>%
  step_unknown(all_nominal_predictors(), new_level = "unknown") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

wf <- workflow() %>%
  add_recipe(rec) %>%
  add_model(spec) %>% add_case_weights(case_wts)

```

```
set.seed(seed)

set.seed(seed)
data_folds <- vfold_cv(data_train, strata = IPAQ_category
                       , v = 5
)

data_folds


library(doParallel)


library(future)
plan(multisession, workers = parallel::detectCores() - 4)


# Bayesian tuning
set.seed(seed)
(start_time <- Sys.time())
res <- tune_bayes(
  wf,
  resamples = data_folds,
  param_info = params,
  initial = 50,
  iter = 20,
  metrics = metric_set(
    yardstick::f_meas,
    yardstick::precision
  ),
  control = control_bayes(
    verbose = TRUE,
    no_improve = 10,
    seed = 123,
    save_pred = TRUE,
    allow_par = TRUE
  )
)
end_time <- Sys.time()
(parallel_time <- end_time - start_time)

ipaq_cat_lgbm_res <- res


res <- ipaq_cat_lgbm_res



cross <- cross %>%
  mutate(case_wts = ifelse(IPAQ_category == "Yes", 1, 2),
         case_wts = importance_weights(case_wts))

set.seed(seed)
```

```
data_split <- initial_split(cross, strata = IPAQ_category, prop = 0.70)
data_train <- training(data_split)
data_test <- testing(data_split)

collect_metrics(res)

best_parms <- select_best(res, metric = "precision")

spec <- boost_tree(
  trees = best_parms$trees,
  tree_depth = best_parms$tree_depth,
  min_n = best_parms$min_n,
  loss_reduction = best_parms$loss_reduction,
  sample_size = best_parms$sample_size,
  learn_rate = best_parms$learn_rate
) %>%
  set_engine("lightgbm",
             lambda_l1 = best_parms$lambda_l1,
             lambda_l2 = best_parms$lambda_l2
             , num_leaves = best_parms$num_leaves) %>%
  set_mode("classification")


rec <- recipe(IPAQ_category ~ ., data = data_train) %>%
  step_unknown(all_nominal_predictors(), new_level = "unknown") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

final <- workflow() %>%
  add_recipe(rec) %>%
  add_model(spec) %>% add_case_weights(case_wts)

set.seed(seed)
final_fit <- fit(final, data = data_train)

final_res <- last_fit(final, data_split, metrics = metric_set(
  yardstick::f_meas,
  yardstick::precision,
  yardstick::recall,
  yardstick::spec,
  yardstick::accuracy,
  yardstick::bal_accuracy,
  yardstick::pr_auc
))

collect_metrics(final_res)

preds <- collect_predictions(final_res) %>%
  mutate(.pred_class = factor(if_else(.pred_Yes >= 0.5, "Yes", "No"), levels =
      c("Yes", "No")))

conf_mat(preds, truth = IPAQ_category, estimate = .pred_class)

label <- 'IPAQ Category'
model <- 'LightGBM'
vip(final_fit, num_features = 10) +
  ggtitle(paste('Most predictive features for\n', label, 'using', model))
```

Longitudinal software code

```
# Software code in Python for the RNN sequence prediction




import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
import itertools as itr
from skimpy import skim
from scipy.stats import iqr
from sklearn.model_selection import train_test_split
from feature_engine.timeseries.forecasting import LagFeatures
from feature_engine.timeseries.forecasting import WindowFeatures
from feature_engine.timeseries.forecasting import ExpandingWindowFeatures
import lightgbm as lgb
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.metrics import median_absolute_error
from sktime.performance_metrics.forecasting import
    MedianAbsolutePercentageError
from sklearn.metrics import mean_absolute_error, median_absolute_error,
    r2_score

import tensorflow as tf
import random



import os
import time
day_number = 7


SEED = 99
tf.random.set_seed(SEED)
random.seed(SEED)
np.random.seed(SEED)

garmin = pd.read_excel('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    Garmin_days_EMA_Anas.xlsx',
                       index_col=0)
ema = pd.read_csv('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    EMA_days_Answered_Final.csv'
                  , sep=';'
                  , decimal=',')

garmin_valid_ids = garmin[garmin['day'] == 14]['participant_id'].unique()

garmin = (garmin
          .query("day <= 14 and participant_id in @garmin_valid_ids"))
```

```python


garmin = (garmin
          .groupby(['participant_id', 'day', 'date', 'hours_cat'])
          .agg(Steps = ("Steps", lambda x: np.sum(x)))
          .sort_values(['participant_id', 'date', 'hours_cat'])
          .reset_index(drop=False))

garmin['hours_cat'] = pd.Categorical(garmin['hours_cat']
     , categories=['Morning', 'Noon', 'Afternoon', 'Evening'])

garmin = (garmin
          .sort_values(['participant_id', 'day', 'date', 'hours_cat']))



participant_id = garmin['participant_id'].unique()
day = np.arange(1, 15)
hours_cat = garmin['hours_cat'].unique()

template = pd.DataFrame(list(itr.product(participant_id, day, hours_cat)),
                        columns=['participant_id', 'day', 'hours_cat'])

template['timestep'] = (template
                        .groupby('participant_id')
                        .cumcount() + 1)

template = pd.merge(template, garmin, on=["participant_id", "day", "hours_cat"]
                    , how='left')

garmin = template.copy()

ema["Time_cat"] = pd.Categorical(ema['Time_cat'],
    categories=['Morning', 'Noon', 'Afternoon', 'Evening'])

ema = (ema
       .rename(columns = {"Time_cat": "hours_cat"}))

garmin = pd.merge(garmin, ema, how='left',
                  on=["participant_id", "day", "hours_cat"])


garmin['date'] = (garmin
                  .groupby(["participant_id", "day"])['date']
                  .transform(lambda x: x.ffill().bfill()))

garmin.columns


garmin = (garmin
          .get(['participant_id', 'day', 'hours_cat', 'timestep', 'date',
                'PHYSICAL_NORM', 'MENTAL_NORM', 'MOTIVATION_NORM', 'EFFICACY_NORM',
                'CONTEXT_NORM', 'Steps']))


```

```
np.random.seed(SEED)
shuffled_ids = np.random.permutation(participant_id)
n = len(shuffled_ids)

train_size = int(np.floor(0.7 * n))
val_size = int(np.floor(0.1 * n))

train_ids = shuffled_ids[:train_size]
val_ids = shuffled_ids[train_size:train_size + val_size]
test_ids = shuffled_ids[train_size + val_size:]

print(len(train_ids), len(val_ids), len(test_ids))
print(sorted(train_ids))
print(sorted(val_ids))
print(sorted(test_ids))




################################################################################
# Yeo-Johnson

from feature_engine.transformation import YeoJohnsonTransformer


steps_train_df = garmin[garmin['participant_id'].isin(train_ids)][['Steps']].
    dropna()
step_transformer = YeoJohnsonTransformer(variables=['Steps'])
step_transformer.fit(steps_train_df)

garmin['Steps_original'] = garmin['Steps']

steps_non_null = garmin.loc[garmin['Steps'].notna(), ['Steps']]
transformed_steps = step_transformer.transform(steps_non_null)

garmin['Steps_transformed'] = np.nan
garmin.loc[steps_non_null.index, 'Steps_transformed'] = transformed_steps['
    Steps']

garmin['Steps'] = garmin['Steps_transformed']




################################################################################
mask = -999
garmin = garmin.fillna(mask)
################################################################################
lable = "Number of Steps"
model = "RNN"

lag_vars = ['Steps'
                , "PHYSICAL_NORM", "MENTAL_NORM", "MOTIVATION_NORM", "
                    EFFICACY_NORM", "CONTEXT_NORM"
]

```

```python
length = 4*day_number
lag_range = np.arange(1, length+1).tolist()


hours_map = {'Morning': 0, 'Noon': 1, 'Afternoon': 2, 'Evening': 3}
garmin['hours_idx'] = garmin['hours_cat'].map(hours_map)

garmin = pd.concat([garmin, pd.get_dummies(garmin['hours_cat'])], axis=1)
garmin[['Morning', 'Noon', 'Afternoon', 'Evening']] = garmin[['Morning', 'Noon
    ', 'Afternoon', 'Evening']].astype(int)


def make_lag(df):
    lf = LagFeatures(periods=lag_range
                     , variables=lag_vars
                     , missing_values='ignore')
    return lf.fit_transform(df)



garmin = (
    garmin
    .groupby(['participant_id'])
    .apply(make_lag)
    .reset_index(drop=True)
    )

garmin.columns





# multi step
for i in range(0, 4):
    garmin[f'Steps_t{i}'] = garmin.groupby('participant_id')['Steps'].shift(-i)
    garmin[f'Steps_original_t{i}'] = garmin.groupby('participant_id')['
        Steps_original'].shift(-i)


target_cols = [f'Steps_t{i}' for i in range(0, 4)]

target_original_cols = [f'Steps_original_t{i}' for i in range(4)]

no_missing = garmin[target_original_cols].notna().all(axis=1)
no_missing = garmin[target_original_cols].notna().all(axis=1)
no_mask = (garmin[target_original_cols] != mask).all(axis=1)

data_train = garmin[
    garmin['participant_id'].isin(train_ids) &
    (garmin['timestep'] > length) &
    no_missing &
    no_mask
]

data_val = garmin[
    garmin['participant_id'].isin(val_ids) &
    (garmin['timestep'] > length) &
    no_missing &
    no_mask
]

data_test = garmin[
    garmin['participant_id'].isin(test_ids) &
    (garmin['timestep'] > length) &
    no_missing &
    no_mask
]


lagged_features = garmin.filter(regex=r"_lag_\d+$").columns.tolist()



other_features = ['hours_cat']
time_of_day_features = ['Noon', 'Afternoon', 'Evening']


features = (time_of_day_features+
            lagged_features)



sorted_lagged_columns = sorted(
    [col for col in data_train.columns if 'Steps_lag_' in col],
    key=lambda x: int(x.split('_')[-1]),
    reverse=True
)



X_train = (data_train
           .get(features #+ ['participant_id']
                ))
y_train = data_train.loc[:, target_cols]

X_val = (data_val
           .get(features #+ ['participant_id']
                ))
y_val = data_val.loc[:, target_cols]


X_test = (data_test
          .get(features #+ ['participant_id']
               ))
y_test = data_test.loc[:, target_cols]





step_cols = [f"Steps_lag_{i}" for i in range(length, 0, -1)]
```

```python
ema_vars = ["PHYSICAL_NORM", "MENTAL_NORM", "MOTIVATION_NORM", "EFFICACY_NORM",
     "CONTEXT_NORM"]
ema_cols = [[f"{var}_lag_{i}" for i in range(length, 0, -1)] for var in
    ema_vars]
time_cols = ["Noon", "Afternoon", "Evening"]

# Train
steps = X_train[step_cols].values.reshape(-1, length, 1)
ema_0 = X_train[ema_cols[0]].values.reshape(-1, length, 1)
ema_1 = X_train[ema_cols[1]].values.reshape(-1, length, 1)
ema_2 = X_train[ema_cols[2]].values.reshape(-1, length, 1)
ema_3 = X_train[ema_cols[3]].values.reshape(-1, length, 1)
ema_4 = X_train[ema_cols[4]].values.reshape(-1, length, 1)
time = X_train[time_cols].values.reshape(-1, 1, 3)
time_repeated = np.repeat(time, length, axis=1)
X_train_seq = np.concatenate([steps
                              #, ema_0, ema_1, ema_2, ema_3, ema_4
                              , time_repeated], axis=2)

# Val
steps = X_val[step_cols].values.reshape(-1, length, 1)
ema_0 = X_val[ema_cols[0]].values.reshape(-1, length, 1)
ema_1 = X_val[ema_cols[1]].values.reshape(-1, length, 1)
ema_2 = X_val[ema_cols[2]].values.reshape(-1, length, 1)
ema_3 = X_val[ema_cols[3]].values.reshape(-1, length, 1)
ema_4 = X_val[ema_cols[4]].values.reshape(-1, length, 1)
time = X_val[time_cols].values.reshape(-1, 1, 3)
time_repeated = np.repeat(time, length, axis=1)
X_val_seq = np.concatenate([steps
                            #, ema_0, ema_1, ema_2, ema_3, ema_4
                            , time_repeated], axis=2)

# Test
steps = X_test[step_cols].values.reshape(-1, length, 1)
ema_0 = X_test[ema_cols[0]].values.reshape(-1, length, 1)
ema_1 = X_test[ema_cols[1]].values.reshape(-1, length, 1)
ema_2 = X_test[ema_cols[2]].values.reshape(-1, length, 1)
ema_3 = X_test[ema_cols[3]].values.reshape(-1, length, 1)
ema_4 = X_test[ema_cols[4]].values.reshape(-1, length, 1)
time = X_test[time_cols].values.reshape(-1, 1, 3)
time_repeated = np.repeat(time, length, axis=1)
X_test_seq = np.concatenate([steps
                             #, ema_0, ema_1, ema_2, ema_3, ema_4
                             , time_repeated], axis=2)





X_train = X_train_seq
X_val = X_val_seq
X_test = X_test_seq



from sklearn.utils import shuffle
```

```

X_train, y_train = shuffle(X_train, y_train, random_state=42)

X_val, y_val = shuffle(X_val, y_val, random_state=42)
X_test, y_test = shuffle(X_test, y_test, random_state=42)






train_2d = X_train.reshape(-1, X_train.shape[-1])
medians = np.median(train_2d, axis=0)
iqrs = np.subtract(*np.percentile(train_2d, [75, 25], axis=0))
iqrs[-4:] = 1.0

iqrs[iqrs == 0] = 1e-8


def robust_scale_ignore_mask(X, medians, iqrs, mask_value=-999):
    mask = (X == mask_value)
    X_masked = np.where(mask, np.nan, X)

    X_scaled = (X_masked - medians) / iqrs

    X_scaled[mask] = mask_value

    return X_scaled


X_train = robust_scale_ignore_mask(X_train, medians, iqrs, mask_value=-999)
X_val = robust_scale_ignore_mask(X_val, medians, iqrs, mask_value=-999)
X_test = robust_scale_ignore_mask(X_test, medians, iqrs, mask_value=-999)


###############################################################################

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import r2_score
from tensorflow.keras.layers import Masking, GRU, Dense

X_train = np.array(X_train)
X_val = np.array(X_val)
X_test = np.array(X_test)

y_train = np.array(y_train)
y_val = np.array(y_val)
y_test = np.array(y_test)

###############################################################################
# modeling

model = Sequential([
    Masking(mask_value=mask, input_shape=(X_train.shape[1], X_train.shape[2])),

    GRU(128, return_sequences=True),
    GRU(64, return_sequences=False),

    Dense(16, activation='relu'),
    Dense(4)
])

model = Sequential([
    Masking(mask_value=mask, input_shape=(X_train.shape[1], X_train.shape[2])),

    LSTM(128, return_sequences=True),

    LSTM(64, return_sequences=False),

    Dense(16, activation='relu'),
    Dense(4)
])


from tensorflow.keras.optimizers import Adam

optimizer = Adam(learning_rate=0.005)

model.compile(optimizer=optimizer, loss='mae', metrics=['mae'])

early_stop = EarlyStopping(monitor='val_loss', patience=100,
    restore_best_weights=True)

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=20,
    batch_size=16,
    callbacks=[early_stop],
    verbose=1
)




y_pred_train = model.predict(X_train)
y_pred_val = model.predict(X_val)
y_pred_test = model.predict(X_test)




def evaluate(y_true, y_pred, name=""):
    #y_true = pd.Series(y_true).reset_index(drop=True)
    #y_pred = pd.Series(y_pred).reset_index(drop=True)

    mae = mean_absolute_error(y_true, y_pred)
    medae = median_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    mean_val = np.mean(y_true)
    median_val = np.median(y_true)



    print(f"\n{name} Set Evaluation:")
    print(f"MAE:            {mae:.2f}")
    print(f"MedAE:          {medae:.2f}")
    print(f"R2:             {r2:.2f}")
    print(f"Mean:           {mean_val:.2f}")
    print(f"Median:         {median_val:.2f}")
    print(f"MAE / Mean:     {mae / mean_val:.3f}")
    print(f"MedAE / Median: {medae / median_val:.3f}")


    return {
            'MAE': round(mae, 2),
            'MedAE': round(medae, 2),
            'R2': round(r2, 2),
            'Mean': round(mean_val, 2),
            'Median': round(median_val, 2),
            'MAE/Mean': round(mae / mean_val, 3),
            'MedAE/Median': round(medae / median_val, 3)
        }



y_train_flat = y_train.reshape(-1)
y_val_flat = y_val.reshape(-1)
y_test_flat = y_test.reshape(-1)

y_pred_train_flat = y_pred_train.reshape(-1)
y_pred_val_flat = y_pred_val.reshape(-1)
y_pred_test_flat = y_pred_test.reshape(-1)



y_train_inv_flat = step_transformer.inverse_transform(pd.DataFrame({'Steps':
    y_train_flat}))['Steps']
y_pred_train_inv_flat = step_transformer.inverse_transform(pd.DataFrame({'Steps
    ': y_pred_train_flat}))['Steps']

y_val_inv_flat = step_transformer.inverse_transform(pd.DataFrame({'Steps':
    y_val_flat}))['Steps']
y_pred_val_inv_flat = step_transformer.inverse_transform(pd.DataFrame({'Steps':
     y_pred_val_flat}))['Steps']

y_test_inv_flat = step_transformer.inverse_transform(pd.DataFrame({'Steps':
    y_test_flat}))['Steps']
y_pred_test_inv_flat = step_transformer.inverse_transform(pd.DataFrame({'Steps
    ': y_pred_test_flat}))['Steps']


# evaluations
evaluate(y_train_inv_flat, y_pred_train_inv_flat, name="Train")
evaluate(y_val_inv_flat, y_pred_val_inv_flat, name="Validation")
evaluate(y_test_inv_flat, y_pred_test_inv_flat, name="Test")
```

```

metrics = evaluate(y_test_inv_flat, y_pred_test_inv_flat, name="Test")

##############################################################################
##############################################################################
##############################################################################
##############################################################################
##############################################################################




import matplotlib.pyplot as plt

# loss curves
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss', linewidth=2)
plt.plot(history.history['val_loss'], label='Validation Loss', linewidth=2)
plt.title('Training and Validation Loss over Epochs')
plt.xlabel('Epoch')
plt.ylabel('MAE Loss')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()




loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 5))
plt.plot(loss, label='Training Loss (MAE)')
plt.plot(val_loss, label='Validation Loss (MAE)')
plt.title('Model Training History')
plt.xlabel('Epoch')
plt.ylabel('MAE')
plt.legend()
plt.tight_layout()
plt.show()

################################################################################
################################################################################
################################################################################
# Code for LightGBM sequence prediction


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
```

```
import itertools as itr
from skimpy import skim
from scipy.stats import iqr
from sklearn.model_selection import train_test_split
from feature_engine.timeseries.forecasting import LagFeatures
from feature_engine.timeseries.forecasting import WindowFeatures
from feature_engine.timeseries.forecasting import ExpandingWindowFeatures
import lightgbm as lgb
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.metrics import median_absolute_error
from sktime.performance_metrics.forecasting import
    MedianAbsolutePercentageError
from sklearn.metrics import mean_absolute_error, median_absolute_error,
    r2_score

import tensorflow as tf
import random



import os
import time
day_number = 7


SEED = 99
tf.random.set_seed(SEED)
random.seed(SEED)
np.random.seed(SEED)

garmin = pd.read_excel('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    Garmin_days_EMA_Anas.xlsx',
                       index_col=0)
ema = pd.read_csv('C:/Users/anasn/Desktop/E/Semester 4/Thesis/files/
    EMA_days_Answered_Final.csv'
                  , sep=';'
                  , decimal=',')

garmin_valid_ids = garmin[garmin['day'] == 14]['participant_id'].unique()

garmin = (garmin
          .query("day <= 14 and participant_id in @garmin_valid_ids"))


garmin = (garmin
          .groupby(['participant_id', 'day', 'date', 'hours_cat'])
          .agg(Steps = ("Steps", lambda x: np.sum(x)))
          .sort_values(['participant_id', 'date', 'hours_cat'])
          .reset_index(drop=False))

garmin['hours_cat'] = pd.Categorical(garmin['hours_cat']
     , categories=['Morning', 'Noon', 'Afternoon', 'Evening'])

garmin = (garmin
          .sort_values(['participant_id', 'day', 'date', 'hours_cat']))
```

```
participant_id = garmin['participant_id'].unique()
day = np.arange(1, 15)
hours_cat = garmin['hours_cat'].unique()

template = pd.DataFrame(list(itr.product(participant_id, day, hours_cat)),
                        columns=['participant_id', 'day', 'hours_cat'])

template['timestep'] = (template
                        .groupby('participant_id')
                        .cumcount() + 1)

template = pd.merge(template, garmin, on=["participant_id", "day", "hours_cat"]
                    , how='left')

garmin = template.copy()

ema["Time_cat"] = pd.Categorical(ema['Time_cat'],
    categories=['Morning', 'Noon', 'Afternoon', 'Evening'])

ema = (ema
       .rename(columns = {"Time_cat": "hours_cat"}))

garmin = pd.merge(garmin, ema, how='left',
                  on=["participant_id", "day", "hours_cat"])


garmin['date'] = (garmin
                  .groupby(["participant_id", "day"])['date']
                  .transform(lambda x: x.ffill().bfill()))

garmin.columns


garmin = (garmin
          .get(['participant_id', 'day', 'hours_cat', 'timestep', 'date',
                'PHYSICAL_NORM', 'MENTAL_NORM', 'MOTIVATION_NORM', '
                    EFFICACY_NORM',
                'CONTEXT_NORM', 'Steps']))







np.random.seed(SEED)
shuffled_ids = np.random.permutation(participant_id)
n = len(shuffled_ids)

train_size = int(np.floor(0.7 * n))
val_size = int(np.floor(0.1 * n))

train_ids = shuffled_ids[:train_size]
```

```
val_ids = shuffled_ids[train_size:train_size + val_size]
test_ids = shuffled_ids[train_size + val_size:]

print(len(train_ids), len(val_ids), len(test_ids))
print(sorted(train_ids))
print(sorted(val_ids))
print(sorted(test_ids))




################################################################################
# Yeo-Johnson

from feature_engine.transformation import YeoJohnsonTransformer


steps_train_df = garmin[garmin['participant_id'].isin(train_ids)][['Steps']].
    dropna()
step_transformer = YeoJohnsonTransformer(variables=['Steps'])
step_transformer.fit(steps_train_df)

garmin['Steps_original'] = garmin['Steps']

steps_non_null = garmin.loc[garmin['Steps'].notna(), ['Steps']]
transformed_steps = step_transformer.transform(steps_non_null)

garmin['Steps_transformed'] = np.nan
garmin.loc[steps_non_null.index, 'Steps_transformed'] = transformed_steps['
    Steps']

garmin['Steps'] = garmin['Steps_transformed']

################################################################################
lable = "Number of Steps"
model = "LGBM"
################################################################################
lag_vars = ['Steps'
                #, "PHYSICAL_NORM", "MENTAL_NORM", "MOTIVATION_NORM", "
                    EFFICACY_NORM", "CONTEXT_NORM"
]

length = 4*day_number
lag_range = np.arange(1, length+1).tolist()


hours_map = {'Morning': 0, 'Noon': 1, 'Afternoon': 2, 'Evening': 3}
garmin['hours_idx'] = garmin['hours_cat'].map(hours_map)

garmin = pd.concat([garmin, pd.get_dummies(garmin['hours_cat'])], axis=1)
garmin[['Morning', 'Noon', 'Afternoon', 'Evening']] = garmin[['Morning', 'Noon
    ', 'Afternoon', 'Evening']].astype(int)




```

```python




def make_lag(df):
    lf = LagFeatures(periods=lag_range
                     #list(range(1, length+1))
                     , variables=lag_vars
                     , missing_values='ignore')
    return lf.fit_transform(df)





garmin = (
    garmin
    .groupby(['participant_id'])
    .apply(make_lag)
    .reset_index(drop=True)
    )

garmin.columns


################################################################################
# multi step
for i in range(0, 4):
    garmin[f'Steps_t{i}'] = garmin.groupby('participant_id')['Steps'].shift(-i)
    garmin[f'Steps_original_t{i}'] = garmin.groupby('participant_id')['
        Steps_original'].shift(-i)



target_cols = [f'Steps_t{i}' for i in range(0, 4)]

target_original_cols = [f'Steps_original_t{i}' for i in range(4)]

no_missing = garmin[target_original_cols].notna().all(axis=1)

data_train = garmin[
    garmin['participant_id'].isin(train_ids) &
    (garmin['timestep'] > length) &
    no_missing
]

data_val = garmin[
    garmin['participant_id'].isin(val_ids) &
    (garmin['timestep'] > length) &
    no_missing
]

data_test = garmin[
    garmin['participant_id'].isin(test_ids) &
    (garmin['timestep'] > length) &
    no_missing
```

```
]






lagged_features = garmin.filter(regex=r"_lag_\d+$").columns.tolist()






other_features = ['hours_cat']
time_of_day_features = ['Noon', 'Afternoon', 'Evening']


features = (time_of_day_features
            + lagged_features)


X_train = (data_train
           .get(features + ['participant_id']))
y_train = data_train.loc[:, target_cols]

X_val = (data_val
           .get(features + ['participant_id']))
y_val = data_val.loc[:, target_cols]


X_test = (data_test
          .get(features + ['participant_id']))
y_test = data_test.loc[:, target_cols]


################################################################################

from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_absolute_error
import lightgbm as lgb



base_model = lgb.LGBMRegressor(
    n_estimators=3000,
    num_leaves=1000,
    max_depth=100,
    min_child_samples=1,
    min_split_gain=0,
    #subsample=1,
    learning_rate=0.005,
    reg_alpha=0.01,
    reg_lambda=0.01,

    objective='regression_l1',

    random_state=123,
    n_jobs=-1,
    verbosity=-1
)



model = MultiOutputRegressor(base_model)

model.fit(X_train, y_train
          )

y_pred_train = model.predict(X_train)
y_pred_val = model.predict(X_val)
y_pred_test = model.predict(X_test)



def evaluate(y_true, y_pred, name=""):
    #y_true = pd.Series(y_true).reset_index(drop=True)
    #y_pred = pd.Series(y_pred).reset_index(drop=True)

    mae = mean_absolute_error(y_true, y_pred)
    medae = median_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    mean_val = np.mean(y_true)
    median_val = np.median(y_true)



    print(f"\n{name} Set Evaluation:")
    print(f"MAE:            {mae:.2f}")
    print(f"MedAE:          {medae:.2f}")
    print(f"R2:             {r2:.2f}")
    print(f"Mean:           {mean_val:.2f}")
    print(f"Median:         {median_val:.2f}")
    print(f"MAE / Mean:     {mae / mean_val:.3f}")
    print(f"MedAE / Median: {medae / median_val:.3f}")


    return {
            'MAE': round(mae, 2),
            'MedAE': round(medae, 2),
            'R2': round(r2, 2),
            'Mean': round(mean_val, 2),
            'Median': round(median_val, 2),
            'MAE/Mean': round(mae / mean_val, 3),
            'MedAE/Median': round(medae / median_val, 3)
        }

y_train_inv = pd.DataFrame()
y_pred_train_inv = pd.DataFrame()
y_val_inv = pd.DataFrame()
y_pred_val_inv = pd.DataFrame()
y_test_inv = pd.DataFrame()
```

```
y_pred_test_inv = pd.DataFrame()

for i, col in enumerate(y_train.columns):
    col_train = pd.DataFrame({'Steps': y_train.iloc[:, i]})
    col_pred_train = pd.DataFrame({'Steps': y_pred_train[:, i]})

    col_val = pd.DataFrame({'Steps': y_val.iloc[:, i]})
    col_pred_val = pd.DataFrame({'Steps': y_pred_val[:, i]})

    col_test = pd.DataFrame({'Steps': y_test.iloc[:, i]})
    col_pred_test = pd.DataFrame({'Steps': y_pred_test[:, i]})

    y_train_inv[f'Steps_t{i+1}'] = step_transformer.inverse_transform(col_train
        )['Steps']
    y_pred_train_inv[f'Steps_t{i+1}'] = step_transformer.inverse_transform(
        col_pred_train)['Steps']

    y_val_inv[f'Steps_t{i+1}'] = step_transformer.inverse_transform(col_val)['
        Steps']
    y_pred_val_inv[f'Steps_t{i+1}'] = step_transformer.inverse_transform(
        col_pred_val)['Steps']

    y_test_inv[f'Steps_t{i+1}'] = step_transformer.inverse_transform(col_test)
        ['Steps']
    y_pred_test_inv[f'Steps_t{i+1}'] = step_transformer.inverse_transform(
        col_pred_test)['Steps']




# Run evaluations
#evaluate(y_train_inv, y_pred_train_inv, name="Train")
#evaluate(y_val_inv, y_pred_val_inv, name="Validation")
evaluate(y_test_inv, y_pred_test_inv, name="Test")
```