Understanding Risk Factors for Mortality Among Older Individuals: A Comprehensive Analysis Using Classical, Ensemble and Deep Learning Models

M.Sc. Capstone Project Report submitted to the

Institute of Mathematics and Applications, Bhubaneswar

in partial fulfillment of the requirements

of the Degree of

Master of Science

in

Mathematics with Data Science

by

Srimanta Ghosh

University Roll No.: **E1773U234013**

under the supervision of

Dr. Rahul Ghosal

(Assistant Professor, Dept. of Epidemiology and Biostatistics, Arnold School Public Health)

University of South Carolina, Columbia, SC

&

Dr. Sudhakar Sahoo

(Associate Professor)

IMA, Bhubaneswar, Odisha

INSTITUTE OF MATHEMATICS AND APPLICATIONS

Bhubaneswar, Odisha - 751029

Dr. Rahul Ghosal
Assistant Professor, Department of Epidemiology and Biostatistics
Arnold School of Public Health, University of South Carolina, Columbia, SC

.

Dr. Sudhakar Sahoo
Associate Professor
Institute of Mathematics and Applications, Bhubaneswar

Date: _____

Supervisor's Certificate

This is to certify that the work presented in this project report entitled "**Understanding Risk Factors for Mortality Among Older Individuals: A Comprehensive Analysis Using Classical, Ensemble, and Deep Learning Models**" by **Srimanta Ghosh**, University Roll Number: **E1773U234013**, is a record of original research/review work carried out by him under our supervision and guidance in partial fulfillment of the requirements for the M.Sc. in Mathematics with Data Science. Neither this project nor any part of it has been submitted for any degree or diploma to any institute or university in India or abroad.

.

Rahul Ghosal

(Supervisor's signature)

(Supervisor's signature)

Declaration

I, **Srimanta Ghosh**, University Roll Number **E1773U234013**, hereby declare that this project entitled **"Understanding Risk Factors for Mortality Among Older Individuals: A Comprehensive Analysis Using Classical, Ensemble, and Deep Learning Models"** represents my original/review work carried out as an M.Sc. in Mathematics with Data Science student of Institute of Mathematics and Applications, Bhubaneswar, and to the best of my knowledge, it is not a complete copy of any previously published work or written by another person, nor has any material been presented for the award of any other degree or diploma of IMA, Bhubaneswar or any other institution.

Any contribution made to this research by others, with whom I have worked at IMA, Bhubaneswar or elsewhere, is explicitly acknowledged in the project. Works of other authors cited in this project have been duly acknowledged in the References section.

Date: _____

_____

(Student's signature)

Acknowledgment

I would like to express my heartfelt gratitude to Institute of Mathematics and Applications, Bhubaneswar, for providing me with the opportunity to undertake my final year M.Sc. Capstone Project, titled "**Understanding Risk Factors for Mortality Among Older Individuals: A Comprehensive Analysis Using Classical, Ensemble, and Deep Learning Models.**" This project has been an immensely valuable experience, contributing significantly to my academic growth and research capabilities.

I am deeply indebted to my external guide, **Dr. Rahul Ghosal** for his exceptional mentorship, insightful feedback, and constant encouragement throughout the course of this project. His expertise and guidance have played a crucial role in shaping the direction and depth of this research.

I would also like to sincerely thank my internal guide, **Dr. Sudhakar Sahoo** for his continuous support, valuable suggestions, and academic guidance that greatly enriched the quality of my work.

Finally, I extend my appreciation to my professors, peers, and everyone who supported me during this project. This research endeavor has been a defining part of my academic journey, and I am confident that the skills and knowledge gained will serve as a strong foundation for my future pursuits in data science and research.

Date: _____

IMA, Bhubaneswar

Srimanta Ghosh
Roll No.: E1773U234013

Abstract

Mortality among older adults remains a pressing public health concern, particularly due to the interplay between chronic health conditions, physical activity, and demographic disparities. While classical survival models such as the Cox proportional hazards regression have long been used to assess mortality risk, they often fall short in capturing the nonlinear and high-dimensional relationships present in real-world health data. With the increasing availability of high-resolution physical activity data from wearable devices and the advancement of machine learning techniques, there is growing potential to enhance mortality prediction and risk stratification.

This project integrates traditional statistical models, ensemble machine learning methods, and deep learning approaches to develop a robust framework for predicting all-cause mortality in older adults. Using data from the 2011–2014 cycles of the National Health and Nutrition Examination Survey (NHANES), which include minute-level accelerometry records from wrist-worn devices linked to mortality outcomes via the National Death Index (NDI), the study incorporates both static features (demographics, comorbidities) and dynamic physical activity profiles. Functional Principal Component Analysis (FPCA) is employed to extract key temporal activity patterns, and these are used as inputs to survival models including Cox PH, Random Survival Forests (RSF), and DeepSurv.

To advance beyond conventional approaches, we also implement a novel BiLSTM-DeepSurv hybrid architecture that directly models smoothed minute-level activity sequences, capturing temporal dependencies in physical behavior. This design enables end-to-end learning from time-series data and offers new insights into how daily activity rhythms relate to survival outcomes.

Objectives

- To explore the relationship between physical activity, demographic factors, and health indicators with mortality in older adults.
- To compare the effectiveness of Cox proportional hazards models, Random Survival Forest (RSF), and deep learning approaches for predicting mortality risk.
- To identify the key risk factors contributing to mortality.

Data Source

The study will use data from the National Health and Nutrition Examination Survey (NHANES) 2011-2014, which includes demographic, lifestyle, and health-related variables. The mortality information is linked to the National Death Index (NDI).

Rahul Ghosal

Signature of The Guide [i]

Signature of the Guide [ii]

Contents

1 Introduction

Aging is an inevitable part of life, but what determines how long and how well we live? For decades, researchers have sought to answer this question, uncovering links between physical activity, health, and longevity. Some studies suggest that simply moving more or exercising can cut the risk of early death by 20-50%[11, 10]. But while we know that staying active is good for us, predicting exactly how lifestyle choices, health conditions and demographic factors to influence mortality still an active research field.

Classical survival analysis methods, such as the Cox proportional hazards model, have long been used to study time-to-event outcomes and identify individual risk factors. While these models are valued for their interpretability, they assume proportional hazards and linear relationships, which may not fully capture the complexities of real-world health data. Ensemble-based models such as Random Survival Forests[5] and deep learning frameworks like DeepSurv[6] offer improved flexibility and predictive power by modeling nonlinear and high-dimensional interactions.

A substantial body of research has applied NHANES accelerometer data to explore mortality and morbidity risk. Most existing studies, however, rely on aggregate measures such as total activity counts, average step counts, or derived metrics like activity-sedentary transition probabilities[14, 8]. More advanced approaches have used functional and distributional analyses to better capture temporal variability; for instance, Cho et al. (2024) found that afternoon activity levels and variability were inversely associated with cardiovascular mortality[2]. Deep learning has also been applied to NHANES data for related tasks, e.g., estimating biological age using convolutional LSTM models[12] or classifying liver disease from clinical features[3]. However, to date, no published study has used BiLSTM-based architectures combined with DeepSurv to directly model minute-level physical activity time series for mortality prediction. This gap represents a novel opportunity to integrate temporal modeling into survival analysis using wearable sensor data.

This project aims to build a comprehensive mortality prediction framework using a combination of classical statistical methods, ensemble learning, and deep learning approaches. Leveraging data from the 2011–2014 NHANES cycles including detailed demographic and clinical variables, as well as minute-level physical activity traces from wrist-worn accelerometers—the study develops and compares several survival models. These include traditional Cox models, penalized and generalized additive Cox models, Random Survival Forests, and neural survival models such as DeepSurv.

To capture the temporal dynamics of daily activity, the study also proposes a novel BiLSTM-DeepSurv hybrid model. This architecture first extracts latent temporal features from raw smoothed activity sequences using bidirectional LSTMs, and then uses these features alongside static variables to predict survival outcomes through a deep Cox model. This approach is designed to leverage the full granularity of time-series activity data without relying solely on summaries like FPCA components and to identify subtle behavioral signals associated with mortality risk.

Objectives

- To explore the relationship between physical activity, demographic factors, and health indicators with mortality in older adults.
- To compare the effectiveness of Cox proportional hazards models, Random Survival Forests (RSF) and Deep learning approaches in predicting mortality risk.
- To identify key static and dynamic risk factors associated with mortality to inform personalized health interventions.

2 Data Description

This study utilizes data from the National Health and Nutrition Examination Survey (NHANES), a large, ongoing, cross-sectional study conducted by the Centers for Disease Control and Preven-

tion (CDC). NHANES employs a multi-stage stratified sampling scheme to collect data on the non-institutionalized US population, covering a broad range of demographic, socioeconomic, lifestyle, and medical factors.

For this research, we focus on a subset of 3,032 individuals (of Age>50 years etc.) from the 2011–2012 and 2013–2014 NHANES waves, where wrist-worn accelerometry data was collected[1]. This dataset is particularly valuable because:

1. It is publicly available and linked to the National Death Index (NDI) by the National Center for Health Statistics (NCHS), allowing for mortality tracking.
2. Physical activity data was collected using wrist-worn accelerometers and processed into Monitor-Independent Movement Summary (MIMS) units, ensuring consistency and reproducibilityici-pants were required to continuously wear the accelerometers for 24 hours, including during sleep, over multiple days, providing a more comprehensive measure of activity levels.

Variables under study:

```
 [1] "SEQN"                   "Age"                    "Race"
 [4] "BMI"                    "sex"                    "Mobility"
 [7] "mortstat"               "diabetes.y"             "poverty_level"
[10] "Asthma"                 "Arthritis"              "heart_failure"
[13] "coronary_heart_disease" "angina"                 "stroke"
[16] "thyroid"                "bronchitis"             "cancer"
[19] "time_mort"              "actmat"
```

- **SEQN:** Unique identifier for each participant in the dataset.
- **Age:** The age of the individual in years.
- **Race:** Racial classification of the individual (1 = Mexican American / 2 = Other Hispanic / 3 = Non-Hispanic White / 4 = Non-Hispanic Black / 6 = Non-Hispanic Asian / 7 = Other Race - Including Multi-Racial)
- **BMI:** Body Mass Index, a measure of body fat based on height and weight.
- **sex:** Biological sex of the individual (0 = Female / 1 = Male).
- **Mobility:** Indicator of mobility.
- **mortstat**: Mortality status (0 = Alive / 1 = Deceased).
- **diabetes.y:** Indicates whether the individual has diabetes (0 = No / 1 = Yes).
- **poverty_level:** Economic status indicator of the individual.
- **Asthma:** Presence of asthma diagnosis (0 = No / 1 = Yes).
- **Arthritis:** Presence of arthritis diagnosis (0 = No / 1 = Yes).
- **heart_failure:** Presence of heart failure diagnosis (0 = No / 1 = Yes).
- **coronary_heart_disease:** Indicates whether the individual has coronary heart disease (0 = No / 1 = Yes).
- **angina:** Presence of angina (chest pain due to heart disease) (0 = No / 1 = Yes).

- **stroke**: Indicates whether the individual has experienced a stroke (0 = No / 1 = Yes).
- **thyroid:** Indicates whether the individual has a thyroid disorder (0 = No / 1 = Yes).
- **bronchitis:** Presence of chronic bronchitis or respiratory disease (0 = No / 1 = Yes).
- **cancer:** Indicates whether the individual has been diagnosed with any type of cancer (0 = No / 1 = Yes).
- **time_mort:** Time (in years) until mortality, or time of follow-up for survival analysis.
- **actmat:** Physical Activity Matrix, summarizing movement levels or exercise frequency.

This dataset provides a unique opportunity to explore the relationship between physical activity, chronic health conditions, and mortality risk. By integrating accelerometry data with health and demographic indicators, this study aims to enhance mortality risk prediction using classical, ensemble, and deep learning models.

3 Survival Analysis

Survival analysis is a statistical framework used to model the time until an event of interest occurs. In the context of the project, the event of interest is mortality among older individuals, and the goal is to predict survival time of an individual and understand how physical activity, demographic factors, and health indicators influence survival time. Unlike traditional regression methods, survival analysis accommodates "censored data", where some individuals are still alive at the time of study completion, meaning their exact survival time is unknown.

Survival and Censoring Times:

Each individual in the dataset has a true survival time T (i.e., time until death) and a censoring time C (i.e., time when the individual was last observed alive). However, we do not always observe T, but rather:

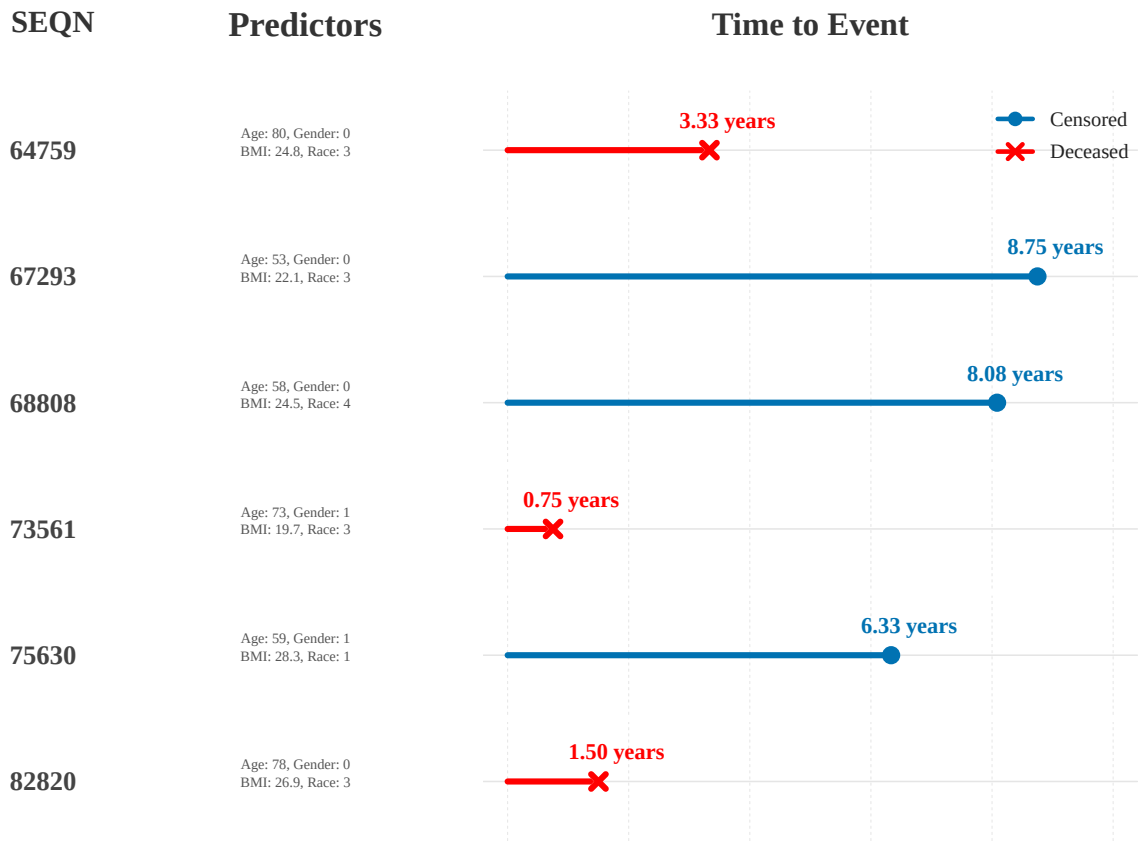
$$Y = \min(T, C)$$

where Y is the observed time, which could either be the actual time of death (T) or the censoring time (C).

We also observe a status indicator:

$$\delta = \begin{cases} 1 & \text{if the event (death) occurred, i.e., } T \leq C \\ 0 & \text{if the event was censored, i.e., } T > C \end{cases}$$

This means that for individuals with $\delta = 0$, their actual survival time is unknown beyond C, making survival analysis necessary.



Survival Function $S(t)$:

The survival function describes the probability that an individual survives beyond a given time t:

$$S(t) = P(T > t)$$

It is a decreasing function, as survival probability naturally declines over time. In our dataset, we aim to estimate $S(t)$ based on factors such as physical activity levels, chronic diseases, and demographic characteristics.

A common non-parametric estimator of $S(t)$ is the Kaplan-Meier estimator, which calculates survival probability step-by-step at observed event times:

$$\hat{S}(t) = \prod_{j:d_j \leq t} \left(\frac{r_j - q_j}{r_j} \right)$$

where:

- d_j = distinct time points when deaths occur
- q_j = number of deaths at d_j
- r_j = number of individuals "at risk" just before d_j

This estimator provides a step-function survival curve, useful for visualizing survival patterns across different groups (e.g., physically active vs. inactive individuals).

Hazard Function $h(t)$:

While $S(t)$ measures survival probability, the hazard function quantifies the instantaneous risk of death at time t, given survival up to t:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t < T \leq t + \Delta t | T > t)}{\Delta t}$$

which simplifies to:

$$h(t) = \frac{f(t)}{S(t)}$$

where:

- $f(t)$ is the probability density function of survival times
- $S(t)$ is the survival function

The cumulative hazard function is given by:

$$H(t) = \int_0^t h(u)\, du = -\log S(t)$$

which helps in modeling survival probabilities in parametric models.

Preprocessing Physical Activity Data:

The raw "actmat" consists of minute-level movement intensity (MIMS units) across 1,440 minutes per day. To smooth it, we apply a 10-minute averaging approach, reducing the matrix from 1,440 to 144 time points per day making it more manageable while preserving meaningful temporal patterns.

Given an individual's activity time series:

$$A_i = \{a_{i1}, a_{i2}, ..., a_{i1440}\}, \quad i = 1, ..., 3032$$

where N is the number of individuals and a_{ij} represents the activity level at minute j for individual i, we define 10-minute intervals as:

$$B_k = \left\{a_{i,(k-1)\times 10+1}, ..., a_{i,k\times 10}\right\}, \quad k = 1, ..., 144$$

The smoothed activity matrix is then computed as:

$$\bar{A}_{ik} = \frac{1}{10} \sum_{j \in B_k} a_{ij}, \quad \forall i, k$$

This transformation results in a new matrix of size 3032×144, where each row represents an individual's daily physical activity curve with 10-minute resolution.

```
Dimensions of actmat: 3032 x 1440
Dimensions of Actsm: 3032 x 144
```

After smoothing, we apply Functional Principal Components Analysis (FPCA)[9] to extract dominant modes of variation in physical activity patterns. FPCA captures underlying movement trends across individuals while reducing dimensionality.

Mathematical Formulation of FPCA

Each individual's daily activity profile, represented as a functional curve, is modeled as:

$$X_i(t) = \mu(t) + \sum_{k=1}^{K} \xi_{ik}\phi_k(t) + \epsilon_i(t)$$

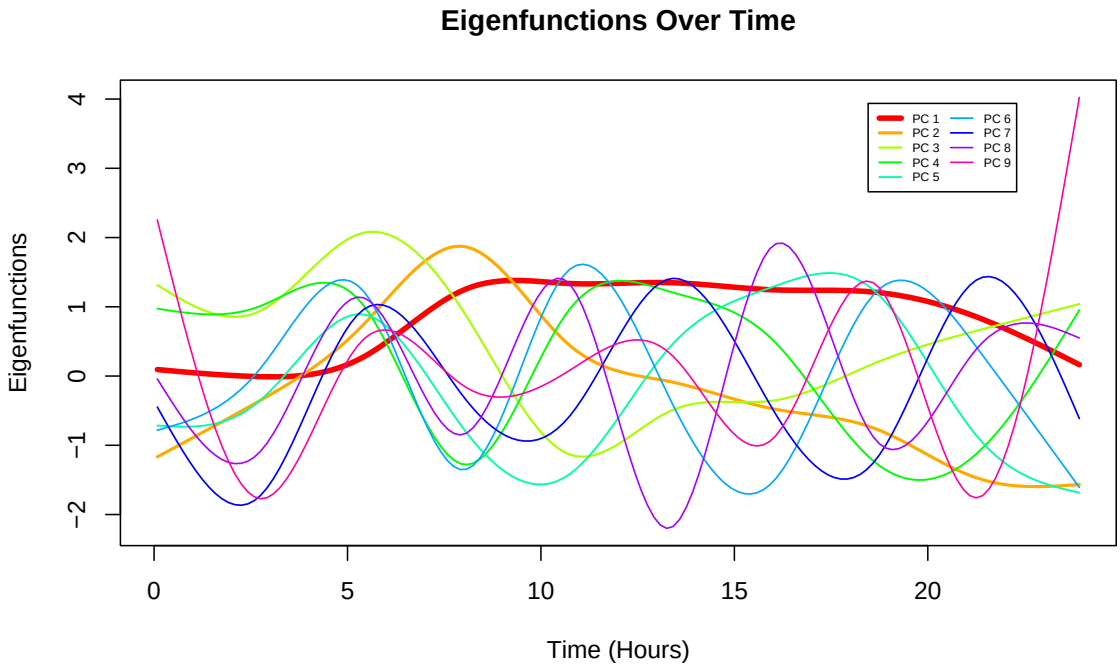
where:

- $X_i(t)$ is the activity function for individual i over time t.
- $\mu(t)$ is the mean activity curve across all individuals.
- $\phi_k(t)$ are the functional principal components (FPCs), capturing dominant activity patterns.
- ξ_{ik} are the individual-specific scores representing how much an individual follows pattern k.
- $\epsilon_i(t)$ is the residual error.

The goal is to retain the first few K components that explain most of the variance:

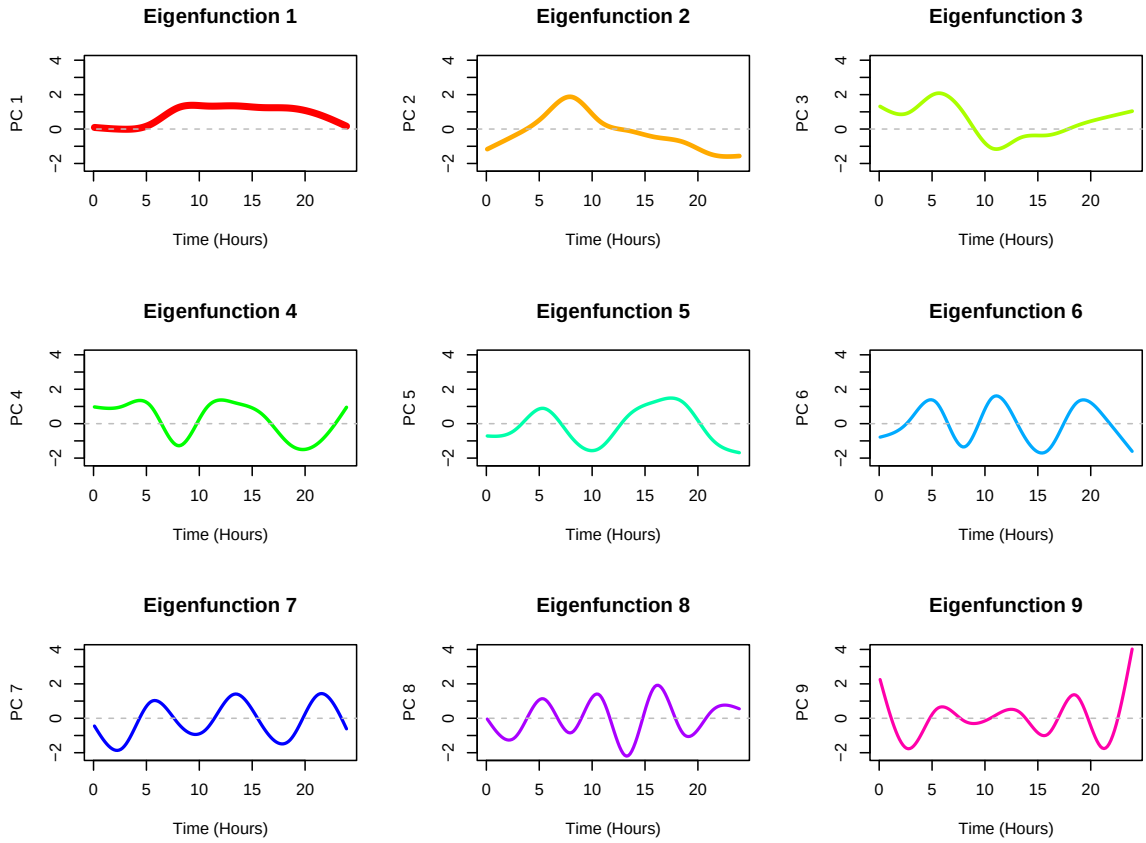
$$\sum_{k=1}^{K} \lambda_k \approx 99\% \text{ of total variance}$$

where λ_k are the eigenvalues associated with $\phi_k(t)$.



Interpretation of FPCA Components

Each functional principal component (FPC) captures a distinct pattern of physical activity behavior:



- **PC1** Accounts for the largest variance in activity patterns. The function remains above zero for most of the day, peaking around 5-15 hours (approximately 5 AM to 3 PM), indicating higher activity during the morning and early afternoon. Individuals with high PC1 scores are likely to have high overall activity levels throughout the day.
- **PC2** has a peak around 6-8 hours (6-8 AM) and decreases afterward. High PC2 scores indicate individuals with early-morning activity spikes, whereas low scores indicate evening-oriented activity.
- **PC3** is more oscillatory, meaning it captures variability in activity levels at different times. There are multiple peaks and troughs, indicating alternating phases of high and low activity.
- **PC4 - PC9** capture finer variations, contributing less than 10% of variance each.

We will analyze whether higher/lower FPCA scores are associated with mortality risk using survival models.

After preprocessing, we introduce and apply survival models, including Cox-PH, RSF, and deep learning-based survival models using FPCA scores as primary exposures and adjusting for other covariates in the study.

Classical Models

Cox Proportional Hazards Model

To assess the impact of physical activity, demographics, and health conditions on survival, we use the Cox Proportional Hazards (Cox-PH) model, which assumes:

$$h(t|X) = h_0(t) \exp(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)$$

where:

- $h_0(t)$ is the baseline hazard function (common to all individuals)
- $X = (X_1, X_2, ..., X_p)$ are predictor variables (e.g., age, activity levels, BMI)
- β_j are coefficients to be estimated, measuring the effect of each predictor on the hazard

This model does not assume a specific form for $h_0(t)$, making it "semi-parametric" and highly flexible for real-world survival data.

Interpreting Cox Model Coefficients

- If $\beta_j > 0$, then increasing X_j increases mortality risk.
- If $\beta_j < 0$, then increasing X_j decreases mortality risk (i.e., protective effect).
- The hazard ratio (HR) is given by:

$$\text{HR} = e^{\beta_j}$$

If HR = 1.2, a one-unit increase in X_j increases the hazard of death by 20%.

In our project, we fitted a Cox-PH model to quantify how physical activity (actmat), chronic diseases, and demographic factors affect mortality risk. We used **stepAIC(cox_model, direction = 'both')** for selecting the best model with relevant predictors.

```
Call:
coxph(formula = Surv(time_mort, mortstat) ~ Age + Race + BMI +
    sex + Mobility + poverty_level + heart_failure + coronary_heart_disease +
    stroke + cancer + PC1 + PC3 + PC9, data = dfrfs)

  n= 3032, number of events= 582

                             coef exp(coef)  se(coef)      z Pr(>|z|)
Age                      0.059707  1.061525  0.005746 10.392  < 2e-16 ***
Race2                   -0.305113  0.737040  0.241007 -1.266  0.20552
Race3                    0.107119  1.113067  0.176779  0.606  0.54455
Race4                   -0.107709  0.897889  0.187611 -0.574  0.56590
Race6                   -0.640421  0.527070  0.272801 -2.348  0.01890 *
Race7                    0.173294  1.189216  0.343155  0.505  0.61356
BMI                     -0.036636  0.964027  0.007270 -5.039 4.67e-07 ***
sex1                    -0.164949  0.847937  0.086623 -1.904  0.05688 .
Mobility2               -0.542814  0.581111  0.096767 -5.609 2.03e-08 ***
poverty_level           -0.081968  0.921302  0.029316 -2.796  0.00517 **
heart_failure1           0.626916  1.871828  0.116411  5.385 7.23e-08 ***
coronary_heart_disease1  0.224023  1.251100  0.114708  1.953  0.05082 .
stroke1                  0.173956  1.190004  0.115436  1.507  0.13182
```

```
cancer1                   0.145309  1.156397  0.093441  1.555  0.11992
PC1                      -0.162558  0.849967  0.021398 -7.597 3.04e-14 ***
PC3                       0.145573  1.156703  0.044577  3.266  0.00109 **
PC9                      -0.179505  0.835684  0.116842 -1.536  0.12446
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

                         exp(coef) exp(-coef) lower .95 upper .95
Age                         1.0615     0.9420    1.0496    1.0735
Race2                       0.7370     1.3568    0.4596    1.1821
Race3                       1.1131     0.8984    0.7871    1.5740
Race4                       0.8979     1.1137    0.6216    1.2969
Race6                       0.5271     1.8973    0.3088    0.8997
Race7                       1.1892     0.8409    0.6070    2.3300
BMI                         0.9640     1.0373    0.9504    0.9779
sex1                        0.8479     1.1793    0.7155    1.0048
Mobility2                   0.5811     1.7208    0.4807    0.7025
poverty_level               0.9213     1.0854    0.8699    0.9758
heart_failure1              1.8718     0.5342    1.4900    2.3516
coronary_heart_disease1     1.2511     0.7993    0.9992    1.5665
stroke1                     1.1900     0.8403    0.9491    1.4921
cancer1                     1.1564     0.8648    0.9629    1.3888
PC1                         0.8500     1.1765    0.8151    0.8864
PC3                         1.1567     0.8645    1.0599    1.2623
PC9                         0.8357     1.1966    0.6646    1.0507

Concordance= 0.783  (se = 0.01 )
Likelihood ratio test= 634.3  on 17 df,   p=<2e-16
Wald test            = 616.4  on 17 df,   p=<2e-16
Score (logrank) test = 706.1  on 17 df,   p=<2e-16
```

The output says, the most significant predictors are "Age", "BMI", "Mobility", "Poverty Level", "heart_failure", "PC1", "PC3" etc.

- $\exp(coef)$, i.e., $H.R.$ for Age is 1.061525, means for each additional year of age, the risk of mortality increases by 6.15%.
- $\exp(-coef)$, i.e., $\frac{1}{H.R.}$ for BMI is 0.9640, means 1-unit increase in BMI reduces the risk of mortality by 3.6%.
- $H.R.$ for heart_failure1 is 1.8718, means a person with a history of heart failure has 87.18% increased morality risk than the person with no history of heart failure.
- PC1 (Overall Activity Level) H.R. = 0.85, i.e., PC1 increases by 1 unit, the hazard of mortality decreases by 15%. More physically active individuals have a lower risk of mortality.
- H.R. for PC3 is 1.157, individuals with higher PC3 scores (who experience greater fluctuations in activity patterns) are at an increased risk of death.

Penalized Cox Model

Group Lasso is an extension of the Lasso regularization technique that selects entire groups of related variables rather than individual predictors. This method is particularly useful when variables naturally belong to predefined groups, such as principal components, categorical dummies etc.

The Cox proportional hazards model when combined with Group Lasso regularization[7], the model reduces overfitting by shrinking coefficients of less relevant groups to zero and handles multicollinearity by retaining or eliminating entire correlated groups together.

Grouping Strategy for Group Lasso

- Continuous variables: Each variable is assigned an independent group.
- Principal components: Treated as a single group to represent the physical activity.
- Binary variables: Each binary variable is assigned an independent group.
- Race: All race dummy variables are grouped together to enable selection or removal as a whole.

```
                   Age                    BMI          poverty_level
          0.0615922635          -0.0292341467          -0.0608897538
                   PC1                    PC2                    PC3
         -0.1284237542           0.0107659177           0.1067255523
                   PC4                    PC5                    PC6
          0.0216365087           0.0368772200           0.0040665728
                   PC7                    PC8                    PC9
         -0.0555885953           0.1055114904          -0.1595919451
                   sex               Mobility              diabetes.y
         -0.1245184918          -0.5620221589           0.0649686194
                Asthma              Arthritis          heart_failure
          0.0000000000          -0.0692911714           0.6272522842
coronary_heart_disease                 angina                 stroke
          0.2093484373           0.0017287661           0.1694214268
               thyroid             bronchitis                 cancer
         -0.0001978324           0.0000000000           0.1165217680
                Race1                  Race2                  Race3
          0.0191559717          -0.1553362257           0.1378523919
                Race4                  Race6                  Race7
         -0.0193857750          -0.3426741922           0.1802807946
```

The coefficients represent the hazard ratios (HRs) for each predictor. Positive values indicate increased mortality risk, while negative values suggest a protective effect. A coefficient of zero implies the variable was not selected by the model.

- Age (0.0616): Older age is associated with a higher mortality risk.
- BMI (-0.0292): Slightly negative, suggesting a person with a higher BMI has lower risk of mortality.
- Poverty Level (-0.0609): A lower poverty level (better economic status) is linked to a reduced mortality risk.
- PC1 (-0.1284) and PC9 (-0.1596): Suggest a protective effect against mortality.
- PC3 (0.1067) and PC5 (0.0369): Indicate increased mortality risk.
- Heart Failure (0.6273) and Coronary Heart Disease (0.2093): Strong risk factors for mortality.
- Stroke (0.1694): Associated with higher mortality risk.
- Cancer (0.1165): Positively linked to increased mortality risk.

- Arthritis (-0.0692): Negative, potentially due to better healthcare engagement.
- Asthma and Bronchitis (0.0000): Not selected, implying no significant association with mortality.
- Sex (-0.1245): Suggests females may have a lower mortality risk compared to males.
- Mobility (-0.5620): Strongly protective; better mobility correlates with lower mortality risk.
- Diabetes (0.0649): A slight increase in mortality risk.
- Thyroid (-0.0002): Minimal impact on mortality.
- Race1 (0.0191), Race2 (-0.1553), Race3 (0.1379), Race4 (-0.0194), Race6 (-0.3426), Race7 (0.1803): The effects vary across racial categories. Some groups (Race6, Race2) have lower risks, while others (Race3, Race7) show increased mortality risks.

Generalized Additive Cox Model

The Generalized Additive Cox Model (GAM-Cox) [4]extends the standard Cox PH model by allowing for non-linear effects of covariates using smooth functions. The hazard function is given by:

$$h(t|X) = h_0(t) \exp\left(\sum_{j=1}^{p} f_j(X_j) + \sum_{k=1}^{q} \beta_k Z_k\right)$$

where:

- $h_0(t)$ is the baseline hazard function.
- $f_j(X_j)$ are smooth, non-linear functions modeled using cubic regression splines ($s(X_j, k, bs =' cr')$).
- Z_k represents categorical variables modeled as linear predictors.
- β_k are regression coefficients for categorical variables.

The penalized likelihood used for estimation is:

$$L(f, \beta) = \prod_{i:E_i=1} \frac{\exp(\sum_{j=1}^{p} f_j(X_{ij}) + \sum_{k=1}^{q} \beta_k Z_{ik})}{\sum_{j \in R(t_i)} \exp(\sum_{j=1}^{p} f_j(X_{ij}) + \sum_{k=1}^{q} \beta_k Z_{ik})} - \lambda \sum_{j=1}^{p} \int \left(f_j''(x)\right)^2 dx$$

where λ is a smoothing parameter controlling model complexity.
The model is implemented using the 'mgcv::gam' function in R:

```
Family: Cox PH
Link function: identity

Formula:
time_mort ~ s(Age, k = 20, bs = "cr") + s(BMI, k = 20, bs = "cr") +
    Race + sex + Mobility + diabetes.y + s(poverty_level, k = 20,
    bs = "cr") + Asthma + Arthritis + heart_failure + coronary_heart_disease +
    angina + stroke + thyroid + bronchitis + cancer + s(PC1,
    k = 20, bs = "cr") + s(PC2, k = 20, bs = "cr") + s(PC3, k = 20,
    bs = "cr") + s(PC4, k = 20, bs = "cr") + s(PC5, k = 20, bs = "cr") +
    s(PC6, k = 20, bs = "cr") + s(PC7, k = 20, bs = "cr") + s(PC8,
    k = 20, bs = "cr") + s(PC9, k = 20, bs = "cr")
```

```
Parametric coefficients:
                        Estimate Std. Error z value Pr(>|z|)
Race2                   -0.24092    0.24438  -0.986   0.3242
Race3                    0.08621    0.18108   0.476   0.6340
Race4                   -0.14632    0.19203  -0.762   0.4461
Race6                   -0.65891    0.27669  -2.381   0.0172 *
Race7                    0.15276    0.34649   0.441   0.6593
sex1                    -0.16392    0.09247  -1.773   0.0763 .
Mobility2               -0.50971    0.09986  -5.104 3.32e-07 ***
diabetes.y1              0.15712    0.09326   1.685   0.0920 .
Asthma1                 -0.03089    0.12110  -0.255   0.7987
Arthritis1              -0.11903    0.08995  -1.323   0.1857
heart_failure1           0.60673    0.11736   5.170 2.34e-07 ***
coronary_heart_disease1  0.20195    0.12311   1.640   0.1009
angina1                  0.08251    0.15773   0.523   0.6009
stroke1                  0.19229    0.11771   1.634   0.1024
thyroid1                -0.04187    0.10999  -0.381   0.7035
bronchitis1              0.03383    0.14693   0.230   0.8179
cancer1                  0.14670    0.09499   1.544   0.1225
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                   edf Ref.df  Chi.sq p-value
s(Age)           2.617  3.240 114.569 < 2e-16 ***
s(BMI)           3.656  4.620  52.389 < 2e-16 ***
s(poverty_level) 2.150  2.667   7.969 0.04740 *
s(PC1)           1.048  1.093  41.826 < 2e-16 ***
s(PC2)           2.655  3.463   2.820 0.46780
s(PC3)           2.854  3.656  14.075 0.00468 **
s(PC4)           2.799  3.659   6.831 0.12265
s(PC5)           1.001  1.002   0.031 0.86303
s(PC6)           1.001  1.001   0.452 0.50192
s(PC7)           1.746  2.253   2.644 0.35268
s(PC8)           1.001  1.003   0.335 0.56385
s(PC9)           1.001  1.001   4.570 0.03257 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Deviance explained =   20%
-REML =   4201  Scale est. = 1         n = 3032
```

Parametric Coefficients:

- Race6 (-0.6589, p=0.0172): A significant protective effect, meaning individuals in this race category have a lower risk of mortality.
- Mobility2 (0.2516, p=3.23e-07): Highly significant and positive, indicating that having mobility issues is strongly associated with higher mortality risk.
- Heart failure (0.6067, p=2.34e-07): A strong positive association, suggesting heart failure significantly increases mortality risk.

- Coronary heart disease (0.2019, p=0.1009): A positive coefficient but not statistically significant at 5% level.
- Other diseases (angina, stroke, thyroid, bronchitis, cancer): None are statistically significant, meaning they do not contribute significantly to mortality risk in this model.

Smooth Terms:

- Age (edf = 2.617, p < 2e-16): Highly significant and non-linear, meaning age strongly impacts mortality, and the relationship is not simply linear.
- BMI (edf = 3.656, p < 2e-16): Strong evidence of a non-linear association with mortality. Suggests BMI influences survival in a complex way (possibly U-shaped).
- Poverty Level (edf = 2.617, p = 0.0474): Significant at the 5% level, indicating poverty level affects survival, though the effect is less strong.
- PC1 (edf = 1.048, p < 2e-16): Highly significant, indicating an important effect on survival.
- PC2-PC9: Mixed significance; only PC3 (p=0.00486) and PC9 (p=0.03257) are statistically significant, while others do not show strong effects.

Ensemble Models

Random Survival Forests (RSF)

Random Survival Forests (RSF)[5] is an ensemble-based method that extends decision trees to handle survival data effectively. The RSF model in this study was implemented using the **randomForestSRC** package in R with the following parameters:

- Number of trees ($ntree$) = 500
- Number of randomly selected predictors at each split ($mtry$) = 3
- Minimum terminal node size ($nodesize$) = 15
- Number of random splits per candidate variable ($nsplit$) = 10

RSF Algorithm Steps

1. Draw $ntree$ bootstrap samples from the original data.
2. Grow a survival tree for each bootstrapped dataset by randomly selecting $mtry$ predictors at each node.
3. Use a log-rank test to determine the best split:

$$L(x,c) = \sum_{i=1}^{N} \frac{(d_{i,1} - Y_{i,1}d_i/Y_i)}{\sqrt{\sum_{i=1}^{N} Y_{i,1}/Y_i(1 - Y_{i,1}/Y_i)(Y_i - d_i)/(Y_i - 1)d_i}}$$

   where $d_{i,1}$ and $Y_{i,1}$ represent the number of deaths and individuals at risk at time t_i in one daughter node.
4. Grow the tree fully, ensuring each terminal node has at least $nodesize = 15$ unique deaths.

5. Compute the ensemble cumulative hazard estimate:

$$\hat{H}_e(t|x_i) = \frac{1}{ntree} \sum_{b=1}^{ntree} \hat{H}_b(t|x_i)$$

using the Nelson-Aalen estimator per tree:

$$\hat{H}_h(t) = \sum_{t_{l,h} \leq t} \frac{d_{l,h}}{Y_{l,h}}$$

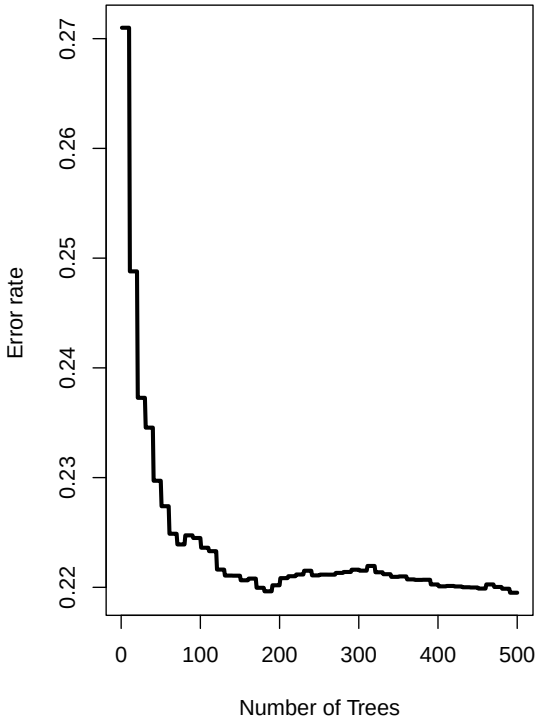
6. Compute feature importance based on the variable permutation method.

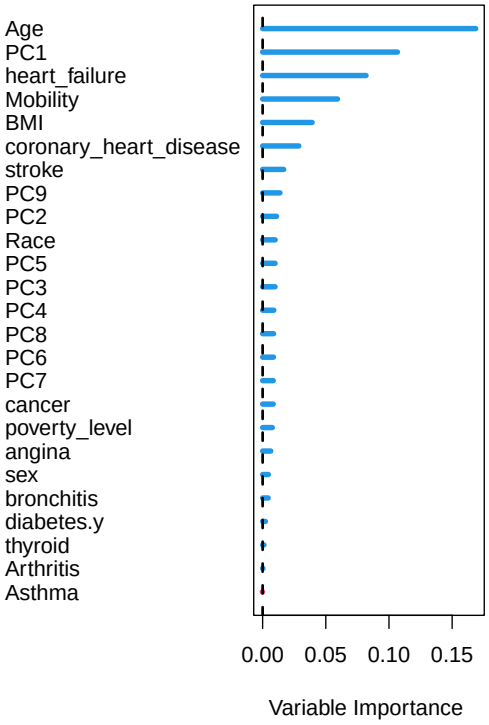
7. Assess the model's predictive performance using the Out-of-Bag (OOB) Concordance Index (C-index).

```
                         Sample size: 3032
                   Number of deaths: 582
                    Number of trees: 500
          Forest terminal node size: 15
       Average no. of terminal nodes: 115.772
No. of variables tried at each split: 3
              Total no. of variables: 25
       Resampling used to grow trees: swor
    Resample size used to grow trees: 1916
                            Analysis: RSF
                              Family: surv
                     Splitting rule: logrank *random*
       Number of random split points: 10
                         (OOB) CRPS: 0.68541513
                  (OOB) stand. CRPS: 0.07833316
   (OOB) Requested performance error: 0.2195079
```

Performance Metrics

- (OOB) Standardized CRPS: 0.0784 is relatively low, meaning the RSF model performs decently.
- (OOB) Requested performance error: 0.2179 suggests reasonable prediction accuracy.





	Importance	Relative Imp
Age	0.1687	1.0000
PC1	0.1067	0.6328
heart_failure	0.0818	0.4853
Mobility	0.0593	0.3514
BMI	0.0392	0.2322
coronary_heart_disease	0.0287	0.1699
stroke	0.0167	0.0989
PC9	0.0137	0.0815
PC2	0.0110	0.0653
Race	0.0099	0.0590
PC5	0.0099	0.0586
PC3	0.0098	0.0583
PC4	0.0089	0.0526
PC8	0.0087	0.0516
PC6	0.0086	0.0511
PC7	0.0084	0.0499
cancer	0.0084	0.0498
poverty_level	0.0077	0.0454
angina	0.0064	0.0382
sex	0.0045	0.0268
bronchitis	0.0044	0.0261
diabetes.y	0.0022	0.0131

```
thyroid                      0.0011          0.0064
Arthritis                    0.0003          0.0018
Asthma                      -0.0003         -0.0017
```

Error Rate vs. Number of Trees Plot

The left-side plot illustrates the model's error rate as a function of the number of trees.

- Initially, the error rate is high when fewer trees are used.
- As the number of trees increases, the error rate decreases and stabilizes.
- Around 300-500 trees, the error rate reaches a plateau, indicating the model has stabilized and additional trees do not significantly improve performance.
- This shows that the ensemble approach is effective and has reached an optimal balance between bias and variance.

Variable Importance Plot

The right-side plot displays the importance of each variable in predicting survival, based on the permutation importance measure.

- Age is the most important predictor, meaning it has the highest impact on survival risk.
- PC1 (Principal Component 1) is the second most influential factor.
- Heart failure, mobility, BMI, and coronary heart disease are among the top risk factors affecting survival.
- Asthma, arthritis, and thyroid disease have the least influence on survival outcomes.

Deep Learning Models

While traditional survival models such as Cox Proportional Hazards, Penalized Cox, GAM Cox, and Random Survival Forest provide strong interpretability and predictive performance, deep learning approaches have emerged as powerful alternatives, especially for capturing complex, high-dimensional relationships in survival data.

DeepSurv

Traditional survival models like the Cox Proportional Hazards (CoxPH) model rely on the proportional hazards assumption and a linear relationship between covariates and the log-risk function. While effective for many applications, these assumptions can be restrictive when the true underlying risk structure is complex or nonlinear—especially in datasets involving interactions between static demographics and high-dimensional behavioral or physiological data (e.g., activity-derived features like PC1–PC9 in our dataset).

To address these limitations, we incorporated DeepSurv, a deep learning-based extension of CoxPH[6], which models the log-risk function $h(x)$ using a feedforward neural network. DeepSurv retains the theoretical advantages of CoxPH (such as handling right-censored data via the partial likelihood) while allowing nonlinear, high-dimensional, and interaction-driven risk modeling without requiring explicit feature engineering.

This approach is particularly suitable for our study, where the goal is to predict all-cause mortality based on both static health indicators (age, BMI, comorbidities, socioeconomic features) and dynamic

physical activity patterns (represented via PC1–PC9). Traditional models may fail to capture the intricate ways these features interact, whereas DeepSurv is capable of learning such relationships from the data.

Model Architecture

DeepSurv is a fully connected feedforward neural network trained to estimate the log-risk function $h_\theta(x)$ using the negative partial log-likelihood from Cox regression as its loss function. The core elements of the model are:

- **Input Layer:** Takes in standardized covariates x (static features and principal components).
- **Hidden Layers:** One or more dense (fully connected) layers with nonlinear activation functions (e.g., ReLU or SELU), optionally followed by dropout to reduce overfitting.
- **Output Layer:** A single linear neuron outputting the log-risk score for each patient.

The output is used within the Cox partial likelihood framework. The model optimizes:

$$\mathcal{L}(\theta) = -\frac{1}{N_{E=1}} \sum_{i:E_i=1} \left[h_\theta(x_i) - \log \sum_{j \in \mathcal{R}(T_i)} \exp(h_\theta(x_j)) \right] + \lambda \|\theta\|_2^2$$

where:

- E_i indicates if an event (death) was observed,
- $\mathcal{R}(T_i)$ is the set of patients still at risk at time T_i,
- λ is the L2 regularization parameter.

Implementation Details

We used the PyCox package in Python to implement DeepSurv. The key steps are as follows:

Data Preprocessing:

- Static continuous features (e.g., age, BMI, poverty level) were standardized.
- Categorical variables (e.g., race, sex, comorbidities) were one-hot encoded.
- Principal component scores (PC1 to PC9), representing time-varying activity data, were included as time-invariant summaries per patient.

Model Configuration:

- Hidden layers: 2
- Neurons per layer: [64, 32]
- Activation function: SELU (self-normalizing)
- Dropout rate: 0.2
- Optimizer: Adam
- Learning rate: 0.001 with inverse time decay
- Regularization: L2 penalty (λ = 1e-4)
- Early stopping and learning rate scheduling were applied to prevent overfitting.

Training Procedure:

- We split the dataset into training and validation sets (80/20).
- The model was trained for up to 1000 epochs with early stopping based on validation concordance.
- Evaluation was conducted using the C-index on a held-out test set.

Limitations

- Although DeepSurv provides greater flexibility and accuracy, it comes at the cost of reduced model interpretability compared to CoxPH.
- Need for large training datasets: Deep learning models are data-hungry and may under perform on small or noisy datasets.
- Tuning sensitivity: Hyperparameter search (number of layers, dropout rate, learning rate) can significantly affect performance.
- Lack of temporal modeling: The current DeepSurv implementation treats PC1–PC9 as time-invariant, though future work could incorporate recurrent or transformer-based architectures for true time-varying analysis.

BiLSTM-based Temporal Feature Extraction Followed by DeepSurv Modeling

While standard DeepSurv models efficiently capture nonlinear relationships between static covariates and survival outcomes, they are not explicitly designed to exploit rich temporal structures present in sequential physiological data such as physical activity traces.

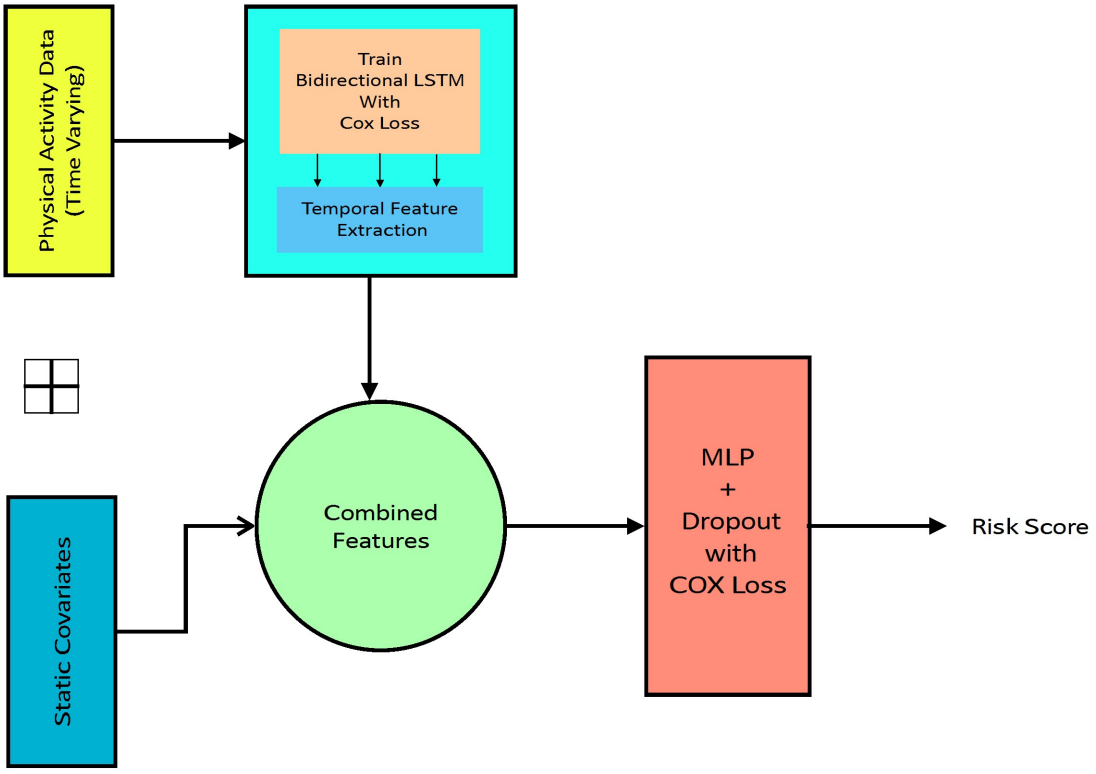
To address this, we implemented a two-stage deep learning framework that integrates temporal feature extraction with deep survival modeling, enhancing the prediction of mortality risk from minute-level accelerometer data.

Motivation and Literature Context

Several recent studies have emphasized the benefits of deep recurrent architectures, such as Long Short-Term Memory (LSTM) networks, in modeling sequential health data[13]. Bidirectional LSTM (BiLSTM) networks, in particular, are capable of capturing forward and backward dependencies within time-series data, making them suitable for analyzing circadian physical activity rhythms. Building upon the DeepSurv framework (Katzman et al., 2018), which adapts the Cox proportional hazards model using feedforward neural networks, we propose a novel hybrid pipeline:

first extracting dynamic latent representations from raw activity sequences via BiLSTM, and then using these representations alongside static demographic and health features to train a DeepSurv model for survival prediction. This approach allows the model to autonomously learn meaningful temporal patterns associated with survival outcomes, without relying solely on handcrafted summaries like functional principal components (PC1–PC9).

Methodology



Data Preprocessing

- Static Features: Age, sex, race, BMI, mobility status, poverty level, presence of chronic diseases (e.g., diabetes, asthma, cancer).
- Time-Varying Features: Raw activity counts ("actmat") across 1440 time points per subject (after smoothing the original 1440-minute recordings).
- Target Variables: Survival time (time_mort) and mortality indicator (mortstat).
- Data were split into training (80%) and validation (20%) cohorts, maintaining consistency with previous modeling stages.

Stage 1: Temporal Feature Learning via BiLSTM

BiLSTM Network Architecture:

- Input: 1440×1 univariate sequence per individual.
- Two LSTM layers, each with 9 hidden units.
- Dropout regularization (0.1) between layers to prevent overfitting.
- Bidirectional processing to capture both morning and evening activity variations.

- Output: Concatenation of final forward and backward hidden states (48-dimensional representation).

CoxPH Loss Training: we trained the BiLSTM jointly with a linear output layer minimizing the negative log partial likelihood of the Cox model. This ensures that the extracted temporal features are optimized for survival discrimination.

Evaluation: The learned representations were validated by computing baseline survival functions and evaluating the time-dependent concordance index (C-index).

Stage 2: DeepSurv Modeling on Extracted Features

Feature Augmentation: The BiLSTM-generated 48-dimensional temporal embeddings were concatenated with standardized static features (age, BMI, comorbidities, etc.) to form the final input for the DeepSurv model.

DeepSurv Network Architecture:

- Hidden Layers: [256, 256] neurons.
- Activation Function: ReLU.
- Dropout: 0.7 after each hidden layer.
- Batch normalization for faster and stable training.

Training Procedure:

- Loss: Negative log partial likelihood of the Cox model.
- Optimizer: Adam with adaptive learning rate tuning.
- Early stopping based on validation C-index.

Results and Discussion

While this setup was designed to take full advantage of both the time-varying and static parts of our data, the actual performance did not surpass some of the simpler models, due to computation limitations. But, even though BiLSTM-DeepSurv didn't outperform the simpler models in this study, it's a promising direction. The approach lets the model learn directly from raw activity data, rather than relying on predefined summaries like PCA. With larger datasets, more refined tuning and more computational resources, this hybrid deep learning model has strong potential for future survival analysis, especially in wearable or sensor-based health monitoring research.

Comparison of Model Performance Using Concordance Index (C-Index)

To see how well different models could predict survival, we used something called the Concordance Index, or C-index. This score tells us how good a model is at ranking people correctly by their risk, in other words, whether it correctly predicts who is more likely to survive longer.

The higher the C-index, the better the model is at making those predictions.

We tested the following models:

- Cox Proportional Hazards Model
- GAM Cox Model (which allows for nonlinear patterns)
- Penalized Cox Model
- Random Survival Forest (RSF)
- DeepSurv (a neural network model)
- BiLSTM-DeepSurv (a deep learning model that handles sequences)

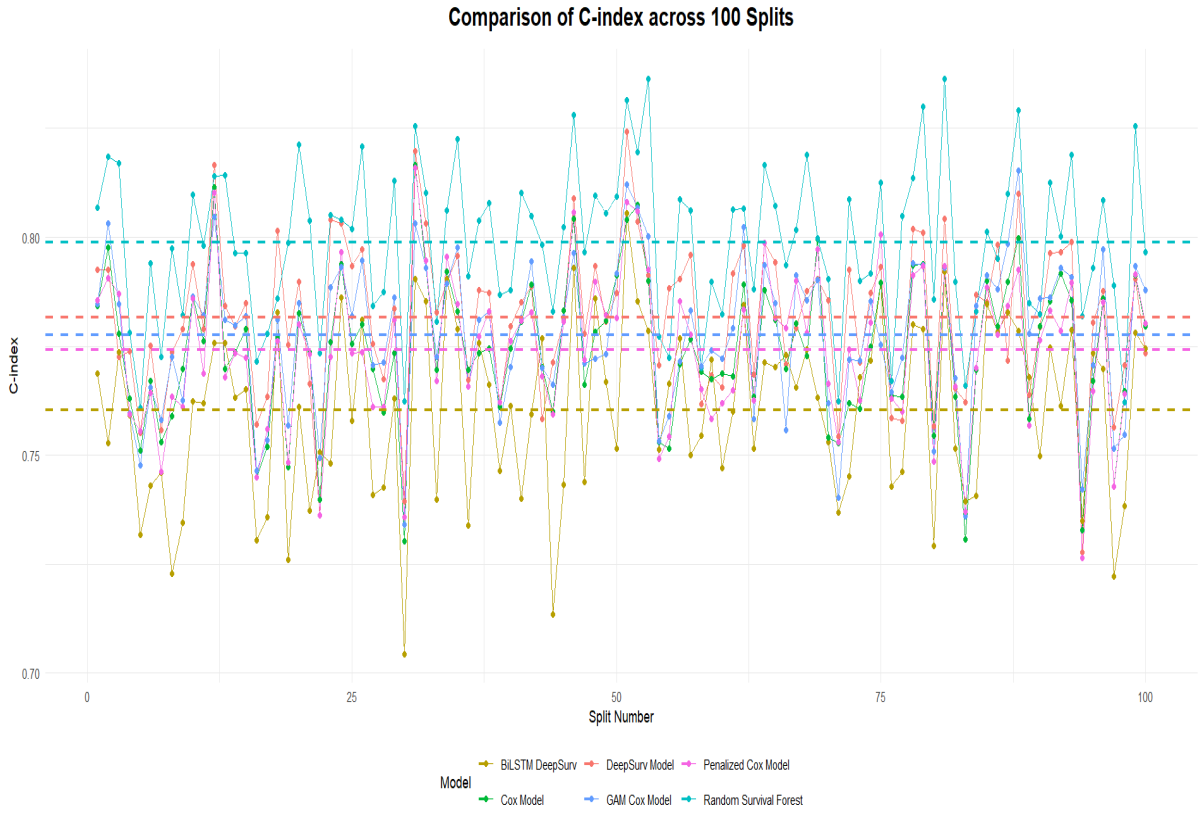
The C-index measures how well the model ranks individuals in terms of survival risk. It is defined as:

$$C = \frac{\sum \mathbf{1}(h(X_i) > h(X_j)) \cdot \mathbf{1}(T_i < T_j)}{\sum \mathbf{1}(T_i < T_j)}$$

where $h(X)$ is the model's predicted risk score, and T is the observed survival time.

To make the comparison fair:

- We split our data into training and testing sets 100 different times (80% training, 20% testing).
- Each model was trained on the training data and tested on the test data.
- We calculated the C-index each time to see how well the models performed.



Mean C-index | DeepSurv: 0.78173 | BiLSTM DeepSurv: 0.76041 | Cox: 0.77421 | RSF: 0.79895 | GAM Cox: 0.77763 | Penalized Cox: 0.77424

Here's a summary of the average C-index for each model:

Model	Average C-Index
Cox	0.77421
GAM Cox	0.77763
Penalized Cox	0.77424
RSF	0.79895
Deepsurv	0.78173
BiLSTM Deepsurv	0.76041

Final Thoughts

- If your main goal is accuracy, go with RSF — it performed the best.
- If you care more about simplicity and interpretability, the Cox models are a solid choice.
- The GAM Cox model gives you a nice middle ground: better predictions than the standard Cox model, and still understandable.
- DeepSurv is a good deep learning option with strong results.
- BiLSTM-DeepSurv is promising, but still needs work — maybe with more computing resources or better tuning, it could do better in future experiments. Or maybe in some other datasets it could perform better.

4 Future work

Several directions can be pursued to build on this work:

- **Improved Temporal Modeling:** Future implementations of the BiLSTM-DeepSurv pipeline could benefit from deeper architectures, attention mechanisms, or transformer-based models that are more effective at capturing long-range dependencies and temporal variability in wearable data.
- **Multi-modal Data Integration:** Incorporating dietary, medication, laboratory, or genomic data from NHANES alongside activity traces could provide a more holistic view of mortality risk factors.
- **Explainability and Interpretability:** Applying model interpretability techniques such as SHAP (SHapley Additive exPlanations) would enhance trust and transparency in healthcare applications.

In conclusion, this project illustrates the value of combining functional data analysis with both classical and modern survival modeling techniques. It also introduces a novel deep learning approach for handling raw time-series physical activity data in mortality prediction. With further methodological refinements and larger datasets, such models hold strong potential for personalized health monitoring and risk stratification in aging populations.

5 Appendix A: R Code

```
############################
## Libraries
############################

library(survival)
library(refund)
library(caret)
library(MASS)
library(grpreg)
library(mgcv)
library(randomForestSRC)
library(ggplot2)
library(scales)
library(patchwork)

############################
## The DataSet
############################

setwd("D:/Final_Year_Project")
load("NHANES_11_14_survPA_TD_Final.RData")
library(dplyr)

time_mort<-baseline$permth_exm/12 #age in month to year-unit
baseline$time_mort<-time_mort
baseline$Age<-baseline$RIDAGEYR
baseline$Race<-baseline$RIDRETH3

df <- baseline %>%
  select(SEQN, Age, Race, BMI, sex, Mobility, mortstat,
         diabetes.y, poverty_level, Asthma,
         Arthritis, heart_failure, coronary_heart_disease, angina, stroke,
         thyroid, bronchitis, cancer, time_mort, actmat) %>%
  filter(!is.na(mortstat) &
           !is.na(time_mort) &
           !is.na(BMI))

c_index_df <- read.csv(file = 'C_index.csv')
names(df)

############################
## A Visualization
############################

# Select relevant data
df_fig <- df %>%
  select(SEQN, time = time_mort, event = mortstat, age = Age, BMI,
         race = Race, gender = sex) %>%
  filter(SEQN %in% c(64759, 67293, 73561, 75630, 68808, 82820))
```

```r
# Custom Colors & Theme
bg_color <- "white"
title_color <- "#333333"
text_color <- "#222222"
event_color <- "red"
censored_color <- "#0072B2"

# Set layout
layout(matrix(c(1:21), 7, 3, byrow = FALSE), widths = c(1, 2, 4))
par(bg = bg_color, family = "serif", col = text_color)

# Title: SEQN
par(mar = c(0, 0, 0, 0))
plot.new()
text(0.5, 0.5, "SEQN", cex = 1.9, font = 2, col = title_color)

# SEQN values
par(mar = c(1, 0, 1, 1))
for(i in 1:6){
  plot.new()
  text(0.5, 0.5, df_fig$SEQN[i], cex = 1.6, font = 2, col = "#444444")
}

# Title: Predictors
par(mar = c(0, 0, 0, 0))
plot.new()
text(0.5, 0.5, "Predictors", cex = 2.2, font = 2, col = title_color)

# Predictor Info
par(mar = c(1, 0, 1, 1))
for(i in 1:6){
  plot.new()
  text(0.5, 0.6, paste0("Age: ", df_fig$age[i],
                        ", Gender: ", df_fig$gender[i],
                        "\nBMI: ", df_fig$BMI[i],
                        ", Race: ", df_fig$race[i]),
       cex = 0.9, font = 1, col = "#555555")
}

# Title: Time to Event
par(mar = c(0, 0, 0, 0))
plot.new()
text(0.5, 0.5, "Time to Event", cex = 2.0, font = 2, col = title_color)

# Survival Data Visualization
par(mar = c(0.2, 0.5, 0.2, 0.5))
for(j in 1:6){
  plot(0, 0, type = "n", xlim = c(0, 10), ylim = c(-1, 1), xaxt = "n",
       yaxt = "n", bty = "n")

  # Gridlines for better visibility
```

```r
  abline(h = 0, col = "gray90", lwd = 1)
  abline(v = seq(0, 10, 2), col = "gray90", lwd = 0.5, lty = 2)

  # Timeline
  segments(0, 0, df_fig$time[j], 0, lwd = 4, col = ifelse(df_fig$event[j] == 1,
                                                          event_color,
                                                          censored_color))

  # Event Marker: Make Deceased (Cross) More Visible
  if(df_fig$event[j] == 1) {
    # White border for better visibility
    points(df_fig$time[j], 0, pch = 4, col = "white", cex = 2, lwd = 4)
    # Bold red cross
    points(df_fig$time[j], 0, pch = 4, col = event_color, cex = 2, lwd = 4)
  } else {
    points(df_fig$time[j], 0, pch = 16, col = censored_color, cex = 2.5)
  }

  # **Adjusted Time Label Placement** (Shifted left & up)
  text(df_fig$time[j] + 0.3, 0.5, paste0(format(round(df_fig$time[j], 2),
                                                nsmall = 2), " years"),
       cex = 1.4, col = ifelse(df_fig$event[j] == 1, event_color,
                               censored_color), font = 2)

  # Legend (only once)
  if(j == 1){
    legend("topright", legend = c("Censored", "Deceased"),
           col = c(censored_color, event_color), pch = c(16, 4), pt.cex = 2,
           lwd = 3, cex = 1.2, bty = "n", y.intersp=1.4)
  }
}

##############################
## FPCA & Data Preprocessing
##############################

tlen <- 10
nt   <- floor(1440/tlen)
tind <- seq(0,1,len=nt)
# create a list of indices for binning into tlen minute windows
inx_col_ls <- split(1:1440, rep(1:nt,each=tlen))
act_mat_bin <- sapply(inx_col_ls, function(x) rowMeans(df$actmat[,x,drop=FALSE],
                                                       na.rm=TRUE))
Y_S<-act_mat_bin
df$Actsm<-Y_S

cat(paste(" Dimensions of actmat:", paste(dim(df$actmat), collapse = " x "),
          "\n","Dimensions of Actsm:", paste(dim(df$Actsm), collapse = " x "),
          "\n"))

fpca_result <- fpca.sc(Y = df$Actsm, pve = 0.99)
```

```r
scoremat<-fpca_result$scores
df$scoremat<-scoremat

tbtick <- seq(0, 1440, by = tlen)
binmid<-tbtick+5
binmid<-binmid[-145]
efmat <- fpca_result$efunctions
explained_variance <- fpca_result$evalues / sum(fpca_result$evalues)
lwd_values <- 1 + 5 * explained_variance

# Plot all eigenfunctions with scaled line thickness
matplot(binmid / 60, efmat, type = "l", lty = 1, col = rainbow(ncol(efmat)),
        lwd = lwd_values, # Use scaled line widths
        xlab = "Time (Hours)", ylab = "Eigenfunctions",
        main = "Eigenfunctions Over Time")

# Add a legend with line thickness explanation
legend(x = "topright", inset = c(0.1, 0.05),
       legend = paste("PC", 1:ncol(efmat)),
       col = rainbow(ncol(efmat)), lty = 1, lwd = lwd_values, cex = 0.5,
       ncol = 2)

# Set up a 2x2 panel layout
par(mfrow = c(3, 3))

# Define colors for consistency
colors <- rainbow(9)

# Compute common y-axis limits
ylim_range <- range(efmat[, 1:9])

# Loop through the first 4 eigenfunctions and plot them separately
for (i in 1:9) {
  plot(binmid / 60, efmat[, i], type = "l", lty = 1, col = colors[i],
       lwd = 2 + 5 * explained_variance[i],
       xlab = "Time (Hours)", ylab = paste("PC", i),
       main = paste("Eigenfunction", i), ylim = ylim_range)
}

dfrfs <- subset(df, select = -c(actmat,Actsm,scoremat))

# Automatically add columns for all principal components
for (i in 1:ncol(scoremat)) {
  dfrfs[[paste0("PC", i)]] <- scoremat[, i]
}
# Columns to convert to numeric
cols_to_convert <- c("Race", "sex", "Mobility")

# Convert specified columns to numeric
dfrfs[cols_to_convert] <- lapply(dfrfs[cols_to_convert], as.factor)
```

```r
##############################
## Cox PH Model
##############################

cox_model <- coxph(Surv(time_mort, mortstat) ~ Age + Race + BMI + sex +
                     Mobility + diabetes.y + poverty_level +
                     Asthma + Arthritis + heart_failure +
                     coronary_heart_disease + angina + stroke + thyroid +
                     bronchitis + cancer  + PC1 + PC2 + PC3 + PC4 + PC5 + PC6 +
                     PC7 + PC8 + PC9, data = dfrfs)

cox_model_subset = stepAIC(cox_model, direction = 'both')
summary(cox_model_subset)

##############################
## Penalized Cox
##############################

df_pencox <- subset(dfrfs, select = -c(SEQN))
race_dummies <- model.matrix(~ Race - 1, data = df_pencox)  # One-hot encoding

binary_vars <- c("sex", "Mobility", "diabetes.y", "Asthma", "Arthritis",
                 "heart_failure", "coronary_heart_disease", "angina", "stroke",
                 "thyroid", "bronchitis", "cancer")

df_pencox[binary_vars] <- lapply(df_pencox[binary_vars],
                                 function(x) as.numeric(as.factor(x)) - 1)

continuous_vars <- c("Age", "BMI", "poverty_level")  # Excluding PC1 to PC9
pc_vars <- c("PC1", "PC2", "PC3", "PC4", "PC5", "PC6", "PC7", "PC8", "PC9")

X <- cbind(df_pencox[, continuous_vars], df_pencox[, pc_vars],
           df_pencox[, binary_vars], race_dummies)

groups <- c(1:length(continuous_vars),  # Continuous variables: separate groups
            rep(length(continuous_vars) + 1, length(pc_vars)),
            (length(continuous_vars) + length(pc_vars) + 1):
              (length(continuous_vars) + length(pc_vars) + length(binary_vars)),
            rep(length(continuous_vars) + length(pc_vars) + length(binary_vars)
                + 1,
                ncol(race_dummies)))  # Race: one group

y <- Surv(df_pencox$time_mort, df_pencox$mortstat)  # Survival outcome

fit <- grpsurv(X, y, group = groups, penalty = "grLasso")

cvfit <- cv.grpsurv(X, y, group = groups, penalty = "grLasso")

# Best lambda value
best_lambda <- cvfit$lambda.min
```

```r
# Coefficients at the optimal lambda
coef(fit, lambda = best_lambda)

################################
## GAM Cox
################################

cox_gam <- mgcv::gam(time_mort ~ s(Age,k=20,bs="cr") + s(BMI,k=20,bs="cr") +
                       Race + sex + Mobility + diabetes.y +
                       s(poverty_level,k=20,bs="cr") +
                       Asthma + Arthritis + heart_failure +
                       coronary_heart_disease + angina + stroke + thyroid +
                       bronchitis + cancer + s(PC1,k=20,bs="cr") +
                       s(PC2,k=20,bs="cr") + s(PC3,k=20,bs="cr") +
                       s(PC4,k=20,bs="cr") + s(PC5,k=20,bs="cr") +
                       s(PC6,k=20,bs="cr") + s(PC7,k=20,bs="cr") +
                       s(PC8,k=20,bs="cr") + s(PC9,k=20,bs="cr"),
                     method = "REML", data = dfrfs, weights = mortstat,
                     family = cox.ph)

summary(cox_gam)

################################
## RSF
################################

rsf_tuned <- rfsrc(Surv(time_mort,mortstat) ~ Age + Race + BMI + sex +
                     Mobility + diabetes.y + poverty_level +
                     Asthma + Arthritis + heart_failure +
                     coronary_heart_disease + angina + stroke + thyroid +
                     bronchitis + cancer  + PC1 + PC2 + PC3 + PC4 + PC5 +
                     PC6 + PC7 + PC8 + PC9,
                   data = dfrfs, ntree = 500, mtry = 3,
                   nodesize = 15, nsplit = 10, importance = TRUE)

print(rsf_tuned)
plot(rsf_tuned)

################################
## 100 Split C-Index Comparison
################################

# Set path to your folder containing the CSV files
data_dir <- "D:/Final_Year_Project/splits_final"

################################
## Cox PH Model
################################

calculate_C_index_cox <- function(n_splits = 100, data_path = "./splits",
                                  train_prefix = "train_split_",
```

```
                                    test_prefix = "test_split_",
                                    factor_cols = c("Race", "sex", "Mobility",
                                                    "diabetes.y", "Asthma",
                                                    "Arthritis",
                                                    "heart_failure",
                                                    "coronary_heart_disease",
                                                    "angina", "stroke",
                                                    "thyroid", "bronchitis",
                                                    "cancer")) {
  c_index_values <- numeric(n_splits)

  for (i in 1:n_splits) {
    train_file <- file.path(data_path, paste0(train_prefix, i, ".csv"))
    test_file  <- file.path(data_path, paste0(test_prefix, i, ".csv"))

    df_train <- read.csv(train_file)
    df_test  <- read.csv(test_file)

    # Convert specified columns to factors in both train and test sets
    for (col in factor_cols) {
      df_train[[col]] <- as.factor(df_train[[col]])
      df_test[[col]]  <- as.factor(df_test[[col]])
    }

    # Fit Cox model
    cox_model <- coxph(Surv(time_mort, mortstat) ~ Age + Race + BMI + sex +
                         Mobility + diabetes.y + poverty_level +
                         Asthma + Arthritis + heart_failure +
                         coronary_heart_disease + angina + stroke + thyroid +
                         bronchitis + cancer + PC1 + PC2 + PC3 + PC4 + PC5 +
                         PC6 + PC7 + PC8 + PC9, data = df_train)

    # Compute risk scores
    risk_cox <- rowSums(predict(cox_model, df_test, type = "terms"))

    # Compute C-index
    c_index_values[i] <- cal_c(marker = risk_cox, Stime = df_test$time_mort,
                               status = df_test$mortstat)
  }

  return(c_index_values)
}

c_index_cox <- calculate_C_index_cox(data_path = data_dir)
mean_c_index_cox <- mean(c_index_cox)

##############################
## Penalized Cox
##############################

calculate_C_index_pencox <- function(n_splits = 100,
```

```
                                                data_path = "./splits",
                                                train_prefix = "train_split_",
                                                test_prefix = "test_split_") {
c_index_values <- numeric(n_splits)

for (i in 1:n_splits) {
  train_file <- file.path(data_path, paste0(train_prefix, i, ".csv"))
  test_file  <- file.path(data_path, paste0(test_prefix, i, ".csv"))

  df_train <- read.csv(train_file)
  df_test  <- read.csv(test_file)

  # One-hot encode Race (drop intercept)
  race_dummies_train <- model.matrix(~ Race - 1, data = df_train)
  race_dummies_test  <- model.matrix(~ Race - 1, data = df_test)

  # Binary variables
  binary_vars <- c("sex", "Mobility", "diabetes.y", "Asthma", "Arthritis",
                   "heart_failure", "coronary_heart_disease", "angina",
                   "stroke","thyroid", "bronchitis", "cancer")

  df_train[binary_vars] <- lapply(df_train[binary_vars], function(x)
    as.numeric(as.factor(x)) - 1)
  df_test[binary_vars]  <- lapply(df_test[binary_vars], function(x)
    as.numeric(as.factor(x)) - 1)

  # Continuous and PC features
  continuous_vars <- c("Age", "BMI", "poverty_level")
  pc_vars <- paste0("PC", 1:9)

  # Design matrices
  X_train <- cbind(df_train[, continuous_vars],
                   df_train[, pc_vars],
                   df_train[, binary_vars],
                   race_dummies_train)

  X_test <- cbind(df_test[, continuous_vars],
                  df_test[, pc_vars],
                  df_test[, binary_vars],
                  race_dummies_test)

  # Group structure
  groups <- c(
    1:length(continuous_vars),  # Continuous: separate groups
    rep(length(continuous_vars) + 1, length(pc_vars)),  # PCs: one group
    (length(continuous_vars) + length(pc_vars) + 1):
      (length(continuous_vars) + length(pc_vars) + length(binary_vars)),
    rep(length(continuous_vars) + length(pc_vars) + length(binary_vars) + 1,
        ncol(race_dummies_train))  # Race: one group
  )
```

```
    # Survival outcomes
    y_train <- Surv(df_train$time_mort, df_train$mortstat)
    y_test  <- Surv(df_test$time_mort, df_test$mortstat)

    # Fit penalized Cox model
    fit <- grpsurv(X_train, y_train, group = groups, penalty = "grLasso")
    cvfit <- cv.grpsurv(X_train, y_train, group = groups, penalty = "grLasso")
    best_lambda <- cvfit$lambda.min

    coefs <- as.numeric(coef(fit, lambda = best_lambda))
    risk_scores <- as.matrix(X_test) %*% coefs

    # Compute C-index
    c_index_values[i] <- cal_c(marker = risk_scores,
                               Stime = df_test$time_mort,
                               status = df_test$mortstat)
  }

  return(c_index_values)
}

c_index_pencox <- calculate_C_index_pencox(data_path = data_dir)
mean_c_index_pencox <- mean(c_index_pencox)

################################
## GAM Cox
################################

calculate_C_index_gamcox <- function(n_splits = 100, data_path = "./splits",
                                     train_prefix = "train_split_",
                                     test_prefix = "test_split_",
                                     factor_cols = c("Race", "sex", "Mobility",
                                                     "diabetes.y", "Asthma",
                                                     "Arthritis",
                                                     "heart_failure",
                                                     "coronary_heart_disease",
                                                     "angina", "stroke",
                                                     "thyroid", "bronchitis",
                                                     "cancer")) {
  c_index_values <- numeric(n_splits)

  for (i in 1:n_splits) {
    # Construct full file paths
    train_file <- file.path(data_path, paste0(train_prefix, i, ".csv"))
    test_file  <- file.path(data_path, paste0(test_prefix, i, ".csv"))

    # Load train and test data
    df_train <- read.csv(train_file)
    df_test  <- read.csv(test_file)

    # Convert specified columns to factors
```

```
    for (col in factor_cols) {
      df_train[[col]] <- as.factor(df_train[[col]])
      df_test[[col]]  <- as.factor(df_test[[col]])
    }

    # Fit GAM Cox model
    cox_gam <- mgcv::gam(time_mort ~ s(Age, k = 20, bs = "cr") +
                           s(BMI, k = 20, bs = "cr") +
                           Race + sex + Mobility + diabetes.y +
                           s(poverty_level, k = 20, bs = "cr") +
                           Asthma + Arthritis + heart_failure +
                           coronary_heart_disease + angina + stroke +
                           thyroid + bronchitis + cancer +
                           s(PC1, k = 20, bs = "cr") +
                           s(PC2, k = 20, bs = "cr") +
                           s(PC3, k = 20, bs = "cr") +
                           s(PC4, k = 20, bs = "cr") +
                           s(PC5, k = 20, bs = "cr") +
                           s(PC6, k = 20, bs = "cr") +
                           s(PC7, k = 20, bs = "cr") +
                           s(PC8, k = 20, bs = "cr") +
                           s(PC9, k = 20, bs = "cr"),
                         method = "REML", data = df_train,
                         weights = mortstat, family = cox.ph)

    # Predict risk scores
    risk_gamcox <- rowSums(predict(cox_gam, df_test, type = "terms"))

    # Compute C-index
    c_index_values[i] <- cal_c(marker = risk_gamcox, Stime = df_test$time_mort,
                               status = df_test$mortstat)
  }

  return(c_index_values)
}

c_index_gamcox <- calculate_C_index_gamcox(data_path = data_dir)
mean_c_index_gamcox <- mean(c_index_gamcox)

################################
## RSF
################################

calculate_C_index_rsf <- function(n_splits = 100, data_path = "./splits",
                                  train_prefix = "train_split_",
                                  test_prefix = "test_split_",
                                  factor_cols = c("Race", "sex", "Mobility",
                                                  "diabetes.y", "Asthma",
                                                  "Arthritis",
                                                  "heart_failure",
                                                  "coronary_heart_disease",
```

```
                                                    "angina", "stroke",
                                                    "thyroid", "bronchitis",
                                                    "cancer")) {
  c_index_values <- numeric(n_splits)

  for (i in 1:n_splits) {
    # File paths
    train_file <- file.path(data_path, paste0(train_prefix, i, ".csv"))
    test_file  <- file.path(data_path, paste0(test_prefix, i, ".csv"))

    # Read train and test data
    df_train <- read.csv(train_file)
    df_test  <- read.csv(test_file)

    # Convert specified columns to factors
    for (col in factor_cols) {
      df_train[[col]] <- as.factor(df_train[[col]])
      df_test[[col]]  <- as.factor(df_test[[col]])
    }

    # Fit RSF model
    rsf_tuned <- rfsrc(Surv(time_mort, mortstat) ~ Age + Race + BMI + sex +
                         Mobility + diabetes.y + poverty_level +
                         Asthma + Arthritis + heart_failure +
                         coronary_heart_disease + angina + stroke + thyroid +
                         bronchitis + cancer + PC1 + PC2 + PC3 + PC4 + PC5 +
                         PC6 + PC7 + PC8 + PC9,
                       data = df_train, ntree = 500, mtry = 3,
                       nodesize = 15, nsplit = 10, importance = TRUE)

    # Determine median death time index for C-index computation
    dis_time <- sort(unique(df_train$time_mort[df_train$mortstat == 1]))
    med_index <- median(1:length(dis_time))

    # Predict survival
    pred <- predict(rsf_tuned, newdata = df_test, times = dis_time, se = FALSE)
    surv_probs <- pred$survival

    # Compute C-index
    surv_obj <- Surv(df_test$time_mort, df_test$mortstat)
    c_index_values[i] <- Cindex(surv_obj, predicted = surv_probs[, med_index])
  }

  return(c_index_values)
}

c_index_rsf <- calculate_C_index_rsf(data_path = data_dir)
mean_c_index_rsf <- mean(c_index_rsf)

results_df <- data.frame(
  c_index_cox = c_index_cox,
```

```
  c_index_rsf = c_index_rsf,
  c_index_gamcox = c_index_gamcox,
  c_index_pencox = c_index_pencox
)

########################################
## Comparison Plot over 100 splits
########################################

# Load data
c_index_df <- read.csv("c_index_all_collected.csv")
c_index_df$Split <- 1:nrow(c_index_df)

# Compute means
mean_c_index_deepsurv <- mean(c_index_df$c_index_deepsurv)
mean_c_index_cox <- mean(c_index_df$c_index_cox)
mean_c_index_rsf <- mean(c_index_df$c_index_rsf)
mean_c_index_gamcox <- mean(c_index_df$c_index_gamcox)
mean_c_index_pencox <- mean(c_index_df$c_index_pencox)
mean_c_index_bilstm <- mean(c_index_df$c_index_bilstmdeepsurv)

# Prepare colors
default_colors <- hue_pal()(6)
model_colors <- c(
  "DeepSurv Model" = default_colors[1],
  "BiLSTM DeepSurv" = default_colors[2],
  "Cox Model" = default_colors[3],
  "Random Survival Forest" = default_colors[4],
  "GAM Cox Model" = default_colors[5],
  "Penalized Cox Model" = default_colors[6]
)

# Main plot
main_plot <- ggplot() +
  geom_line(data = c_index_df, aes(x = Split, y = c_index_deepsurv,
                                   color = "DeepSurv Model")) +
  geom_line(data = c_index_df, aes(x = Split, y = c_index_bilstmdeepsurv,
                                   color = "BiLSTM DeepSurv")) +
  geom_line(data = c_index_df, aes(x = Split, y = c_index_cox,
                                   color = "Cox Model")) +
  geom_line(data = c_index_df, aes(x = Split, y = c_index_rsf,
                                   color = "Random Survival Forest")) +
  geom_line(data = c_index_df, aes(x = Split, y = c_index_gamcox,
                                   color = "GAM Cox Model")) +
  geom_line(data = c_index_df, aes(x = Split, y = c_index_pencox,
                                   color = "Penalized Cox Model")) +

  geom_point(data = c_index_df, aes(x = Split, y = c_index_deepsurv,
                                    color = "DeepSurv Model")) +
  geom_point(data = c_index_df, aes(x = Split, y = c_index_bilstmdeepsurv,
                                    color = "BiLSTM DeepSurv")) +
```

```
  geom_point(data = c_index_df, aes(x = Split, y = c_index_cox,
                                    color = "Cox Model")) +
  geom_point(data = c_index_df, aes(x = Split, y = c_index_rsf,
                                    color = "Random Survival Forest")) +
  geom_point(data = c_index_df, aes(x = Split, y = c_index_gamcox,
                                    color = "GAM Cox Model")) +
  geom_point(data = c_index_df, aes(x = Split, y = c_index_pencox,
                                    color = "Penalized Cox Model")) +

  geom_hline(aes(yintercept = mean_c_index_deepsurv,
                 color = "DeepSurv Model"), linetype = "dashed", size = 1) +
  geom_hline(aes(yintercept = mean_c_index_bilstm,
                 color = "BiLSTM DeepSurv"), linetype = "dashed", size = 1) +
  geom_hline(aes(yintercept = mean_c_index_cox,
                 color = "Cox Model"), linetype = "dashed", size = 1) +
  geom_hline(aes(yintercept = mean_c_index_rsf,
                 color = "Random Survival Forest"), linetype = "dashed",
             size = 1) +
  geom_hline(aes(yintercept = mean_c_index_gamcox,
                 color = "GAM Cox Model"), linetype = "dashed", size = 1) +
  geom_hline(aes(yintercept = mean_c_index_pencox,
                 color = "Penalized Cox Model"), linetype = "dashed",
             size = 1) +

  scale_color_manual(values = model_colors) +
  labs(
    x = "Split Number",
    y = "C-index",
    color = "Model"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 12, face = "bold"),
    legend.position = "bottom"
  )

# Title plot
title_plot <- ggplot() +
  annotate("text", x = 0.5, y = 0.5,
           label = "Comparison of C-index across 100 Splits", size = 6,
           fontface = "bold") +
  theme_void()

# Bottom caption with updated means
mean_label <- paste(
  "Mean C-index | DeepSurv:", round(mean_c_index_deepsurv, 5),
  "| BiLSTM DeepSurv:", round(mean_c_index_bilstm, 5),
  "| Cox:", round(mean_c_index_cox, 5),
  "| RSF:", round(mean_c_index_rsf, 5),
  "| GAM Cox:", round(mean_c_index_gamcox, 5),
  "| Penalized Cox:", round(mean_c_index_pencox, 5)
```

```
)

bottom_caption <- ggplot() +
  annotate("text", x = 0.5, y = 0.5, label = mean_label, size = 4,
           hjust = 0.5) +
  theme_void()

# Combine all
final_plot <- title_plot / main_plot / bottom_caption +
  plot_layout(heights = c(0.08, 1, 0.1))
print(final_plot)
```

6 Appendix B: Python Code

[ ]:
```
# ============================
# 1. DeepSurv over 100 Splits
# ============================
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn_pandas import DataFrameMapper
import torch
import torch.nn as nn
import torchtuples as tt
from pycox.models import CoxPH
from pycox.evaluation import EvalSurv
import os

# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Fixed columns
cols_standardize = ['Age','BMI', 'poverty_level',␣
 ↪'PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9']
cols_leave = ['Race', 'sex', 'Mobility', 'diabetes.y', 'Asthma', 'Arthritis',␣
 ↪'heart_failure', 'coronary_heart_disease',
              'angina', 'stroke', 'thyroid', 'bronchitis', 'cancer']

standardize = [([col], StandardScaler()) for col in cols_standardize]
leave = [(col, None) for col in cols_leave]
x_mapper = DataFrameMapper(standardize + leave)

# Model architecture
def get_model(input_dim):
    num_nodes = [256, 256]
    net = tt.practical.MLPVanilla(input_dim, num_nodes, 1, batch_norm=True,␣
 ↪dropout=0.7, output_bias=False).to(device)
    return CoxPH(net, tt.optim.Adam)

# For C-index storage
c_index_list = []

# Directory containing the splits
split_dir = "D:/Final_Year_Project/splits_final/"  # Modify this if needed

for i in range(1, 101):
    print(f"Processing split {i}...")

    # Load train/test split
    df_train = pd.read_csv(os.path.join(split_dir, f"train_split_{i}.csv"))
    df_test = pd.read_csv(os.path.join(split_dir, f"test_split_{i}.csv"))

    # Fit transformer only on training data
```

```
    x_train = torch.tensor(x_mapper.fit_transform(df_train).astype('float32')).
↪to(device)
    x_val = torch.tensor(x_mapper.transform(df_test).astype('float32')).to(device)
    x_test = torch.tensor(x_mapper.transform(df_test).astype('float32')).
↪to(device)

    # Get target
    get_target = lambda df: (
        torch.tensor(df['time_mort'].values, dtype=torch.float32).to(device),
        torch.tensor(df['mortstat'].values, dtype=torch.float32).to(device)
    )
    y_train = get_target(df_train)
    y_val = get_target(df_test)
    durations_test, events_test = get_target(df_test)
    val = x_val, y_val

    durations_np = durations_test.cpu().numpy()
    events_np = events_test.cpu().numpy()

    # Model
    model = get_model(x_train.shape[1])

    # Find LR
    batch_size = 64
    lrfinder = model.lr_finder(x_train, y_train, batch_size, tolerance=10)
    best_lr = lrfinder.get_best_lr()
    model.optimizer.set_lr(best_lr)

    # Train
    callbacks = [tt.callbacks.EarlyStopping()]
    model.fit(x_train, y_train, batch_size, 512, callbacks, verbose=False,
              val_data=val, val_batch_size=batch_size)

    # Predict survival
    model.compute_baseline_hazards()
    surv = model.predict_surv_df(x_test)
    ev = EvalSurv(surv, durations_np, events_np, censor_surv='km')
    c_index = ev.concordance_td()
    c_index_list.append(c_index)

# Save C-indices
c_index_df = pd.DataFrame({'split': list(range(1, 101)), 'c_index': c_index_list})

# ========================================================
# 2. Train BiLSTM with CoxPH Loss for Feature Extraction
# ========================================================

config = {
    'lstm_hidden_size': 5,
    'lstm_layers': 3,
    'lstm_dropout': 0.3,
```

```
    'batch_size': 64,
    'epochs_lstm_cox': 512,
}

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

def to_device(x):
    if isinstance(x, (tuple, list)):
        return tuple(to_device(xx) for xx in x)
    return x.to(device)

data = pd.read_csv("D:\\Final_Year_Project\\NHANES_11_14_survPA_Python.csv")
data_act = pd.read_csv("D:\\Final_Year_Project\\activity.csv")
df = pd.concat([data, data_act], axis=1)
df['row_id'] = df.index

# Shuffle and split
df = df.sample(frac=1, random_state=42).reset_index(drop=True)
df_val = df.sample(frac=0.1, random_state=42).reset_index(drop=True)
df_train = df.drop(df_val.index).reset_index(drop=True)

static_cols = ['Age', 'Race', 'BMI', 'sex', 'Mobility', 'diabetes.y',␣
↪'poverty_level',
               'Asthma', 'Arthritis', 'heart_failure', 'coronary_heart_disease',
               'angina', 'stroke', 'thyroid', 'bronchitis', 'cancer']
time_cols = [str(i) for i in range(1, 1441)]

def prepare_data(df):
    X_static = df[static_cols].copy()
    X_time = df[time_cols].values.reshape(len(df), 1440, 1)
    y_event = df['mortstat'].values.astype(np.float32)
    y_time = df['time_mort'].values.astype(np.float32)
    row_ids = df['row_id'].values
    return X_static, X_time.astype(np.float32), y_time, y_event, row_ids

X_train_static, X_train_time, time_train, y_train_event, row_ids_train =␣
↪prepare_data(df_train)
X_val_static, X_val_time, time_val, y_val_event, row_ids_val =␣
↪prepare_data(df_val)

class BiLSTMFeatureExtractor(nn.Module):
    def __init__(self, input_size=1, hidden_size=9, num_layers=2, dropout=0.1):
        super().__init__()
        self.lstm = nn.LSTM(input_size=input_size,
                            hidden_size=hidden_size,
                            num_layers=num_layers,
                            dropout=dropout,
                            batch_first=True,
                            bidirectional=True)

    def forward(self, x):
```

```
        _, (hn, _) = self.lstm(x)
        hn_forward = hn[-2]
        hn_backward = hn[-1]
        return torch.cat((hn_forward, hn_backward), dim=1)

class FeatureWrapperForCox(nn.Module):
    def __init__(self, lstm_model, static_input_dim, hidden_dim):
        super().__init__()
        self.lstm_model = lstm_model
        self.linear = nn.Linear(2 * hidden_dim + static_input_dim, 1)

    def forward(self, X_time, X_static):
        lstm_feat = self.lstm_model(X_time)
        return self.linear(torch.cat([lstm_feat, X_static], dim=1))


def train_bilstm_cox(X_train_time, X_train_static, durations_train, events_train,
                     X_val_time, X_val_static, durations_val, events_val):

    scaler = StandardScaler()
    X_train_static = scaler.fit_transform(X_train_static).astype(np.float32)
    X_val_static = scaler.transform(X_val_static).astype(np.float32)

    X_train_time, X_train_static, durations_train, events_train = to_device((
        torch.tensor(X_train_time), torch.tensor(X_train_static),
        torch.tensor(durations_train), torch.tensor(events_train)))
    X_val_time, X_val_static, durations_val, events_val = to_device((
        torch.tensor(X_val_time), torch.tensor(X_val_static),
        torch.tensor(durations_val), torch.tensor(events_val)))

    y_train = (durations_train, events_train)
    y_val = (durations_val, events_val)

    lstm_model = BiLSTMFeatureExtractor(
        hidden_size=config['lstm_hidden_size'],
        num_layers=config['lstm_layers'],
        dropout=config['lstm_dropout']
    ).to(device)

    feature_net = FeatureWrapperForCox(
        lstm_model, X_train_static.shape[1], config['lstm_hidden_size']
    ).to(device)

    model = CoxPH(feature_net, tt.optim.Adam)
    batch_size = config['batch_size']
    model.optimizer.set_lr(model.lr_finder((X_train_time, X_train_static),␣
↪y_train, batch_size=batch_size).get_best_lr())

    model.fit((X_train_time, X_train_static), y_train, batch_size,␣
↪epochs=config['epochs_lstm_cox'],
              callbacks=[tt.callbacks.EarlyStopping()],
```

```
            val_data=((X_val_time, X_val_static), y_val),␣
↪val_batch_size=batch_size)

    model.compute_baseline_hazards()
    surv = model.predict_surv_df((X_val_time, X_val_static))
    ev = EvalSurv(surv, durations_val.cpu().numpy(), events_val.cpu().numpy(),␣
↪censor_surv='km')
    print("BiLSTM-CoxPH Concordance Index:", ev.concordance_td())

    return lstm_model, scaler

def extract_lstm_only_features(lstm_model, X_time):
    X_time = torch.tensor(X_time).to(device)
    with torch.no_grad():
        lstm_feat = lstm_model(X_time)
    return lstm_feat.cpu().numpy()

lstm_model, scaler = train_bilstm_cox(X_train_time, X_train_static, time_train,␣
↪y_train_event,
                                      X_val_time, X_val_static, time_val,␣
↪y_val_event)

train_lstm_features = extract_lstm_only_features(lstm_model, X_train_time)
val_lstm_features = extract_lstm_only_features(lstm_model, X_val_time)

# ====================================
# 3. BiLSTM + DeepSurv over 100 Splits
# ====================================
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn_pandas import DataFrameMapper
import torch
import torchtuples as tt
from pycox.models import CoxPH
from pycox.evaluation import EvalSurv
import os

# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Fixed columns
cols_standardize = ['Age','BMI', 'poverty_level', 'dyn_feat_0', 'dyn_feat_1',␣
↪'dyn_feat_2','dyn_feat_3', 'dyn_feat_4',
                    'dyn_feat_5', 'dyn_feat_6', 'dyn_feat_7','dyn_feat_8',␣
↪'dyn_feat_9']
cols_leave = ['Race', 'sex', 'Mobility', 'diabetes.y', 'Asthma', 'Arthritis',␣
↪'heart_failure', 'coronary_heart_disease',
              'angina', 'stroke', 'thyroid', 'bronchitis', 'cancer']
```

```
standardize = [([col], StandardScaler()) for col in cols_standardize]
leave = [(col, None) for col in cols_leave]
x_mapper = DataFrameMapper(standardize + leave)

# Model architecture
def get_model(input_dim):
    num_nodes = [256, 256]
    net = tt.practical.MLPVanilla(input_dim, num_nodes, 1, batch_norm=True,␣
 ↪dropout=0.7, output_bias=False).to(device)
    return CoxPH(net, tt.optim.Adam)

# For C-index storage
c_index_list = []

# Directory containing the splits
split_dir = "D:/Final_Year_Project/splits_final/"  # Modify this if needed

for i in range(1, 101):
    print(f"Processing split {i}...")

    # Load train/test split
    df_train = pd.read_csv(os.path.join(split_dir, f"train_split_{i}.csv"))
    df_test = pd.read_csv(os.path.join(split_dir, f"test_split_{i}.csv"))

    # Fit transformer only on training data
    x_train = torch.tensor(x_mapper.fit_transform(df_train).astype('float32')).
 ↪to(device)
    x_val = torch.tensor(x_mapper.transform(df_test).astype('float32')).to(device)
    x_test = torch.tensor(x_mapper.transform(df_test).astype('float32')).
 ↪to(device)

    # Get target
    get_target = lambda df: (
        torch.tensor(df['time_mort'].values, dtype=torch.float32).to(device),
        torch.tensor(df['mortstat'].values, dtype=torch.float32).to(device)
    )
    y_train = get_target(df_train)
    y_val = get_target(df_test)
    durations_test, events_test = get_target(df_test)
    val = x_val, y_val

    durations_np = durations_test.cpu().numpy()
    events_np = events_test.cpu().numpy()

    # Model
    model = get_model(x_train.shape[1])

    # Find LR
    batch_size = 64
    lrfinder = model.lr_finder(x_train, y_train, batch_size, tolerance=10)
    best_lr = lrfinder.get_best_lr()
```

```
    model.optimizer.set_lr(best_lr)

    # Train
    callbacks = [tt.callbacks.EarlyStopping()]
    model.fit(x_train, y_train, batch_size, 512, callbacks, verbose=False,
              val_data=val, val_batch_size=batch_size)

    # Predict survival
    model.compute_baseline_hazards()
    surv = model.predict_surv_df(x_test)
    ev = EvalSurv(surv, durations_np, events_np, censor_surv='km')
    c_index = ev.concordance_td()
    c_index_list.append(c_index)

# Save C-indices
c_index_df = pd.DataFrame({'split': list(range(1, 101)), 'c_index': c_index_list})
```

References

[1] Centers for Disease Control and Prevention (CDC) and National Center for Health Statistics (NCHS). National health and nutrition examination survey (2011–2014) data. https://wwwn.cdc.gov/nchs/nhanes/Default.aspx, 2014. Accessed: 2025-01-08.

[2] Sunwoo Emma Cho, Enakshi Saha, Marcos Matabuena, Jingkai Wei, and Rahul Ghosal. Exploring the association between daily distributional patterns of physical activity and cardiovascular mortality risk among older adults in nhanes 2003-2006. *Annals of Epidemiology*, 99:24–31, 2024.

[3] R Geetha, D Jenila Rani, and K Anbarasu. Applying deep learning methods to non-alcoholic fatty liver disease management. In *2024 5th International Conference on Circuits, Control, Communication and Computing (I4C)*, pages 282–286. IEEE, 2024.

[4] TJ Hastie. *Generalized Additive Models*. Routledge, 2017.

[5] Hemant Ishwaran and Udaya B Kogalur. Random survival forests for r. *R news*, 7(2):25–31, 2007.

[6] Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18:1–12, 2018.

[7] Jinseog Kim, Insuk Sohn, Sin-Ho Jung, Sujong Kim, and Changyi Park. Analysis of survival data with group lasso. *Communications in Statistics-Simulation and Computation*, 41(9):1593–1605, 2012.

[8] Lily Koffman and John Muschelli. Minute level step counts and physical activity data from the national health and nutrition examination survey (nhanes) 2011-2014.

[9] Piotr Kokoszka and Matthew Reimherr. *Introduction to functional data analysis*. Chapman and Hall/CRC, 2017.

[10] William E Kraus, Kenneth E Powell, William L Haskell, Kathleen F Janz, Wayne W Campbell, John M Jakicic, Richard P Troiano, Kyle Sprow, Andrea Torres, Katrina L Piercy, et al. Physical activity, all-cause and cardiovascular mortality, and cardiovascular disease. *Medicine and science in sports and exercise*, 51(6):1270, 2019.

[11] Marc Nocon, Theresa Hiemann, Falk Müller-Riemenschneider, Frank Thalau, Stephanie Roll, and Stefan N Willich. Association of physical activity with all-cause and cardiovascular mortality: a systematic review and meta-analysis. *European Journal of Preventive Cardiology*, 15(3):239–246, 2008.

[12] Syed Ashiqur Rahman and Donald A Adjeroh. Deep learning using convolutional lstm estimates biological age from physical activity. *Scientific reports*, 9(1):11425, 2019.

[13] Zhang Runquan and Shi Xiaoping. Lstm-cox model: A concise and efficient deep learning approach for handling recurrent events. *arXiv e-prints*, pages arXiv–2405, 2024.

[14] Daniela Schmid, Cristian Ricci, and Michael F Leitzmann. Associations of objectively assessed physical activity and sedentary time with all-cause mortality in us adults: the nhanes study. *PloS one*, 10(3):e0119591, 2015.