Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07263-9

ORIGINAL ARTICLE

Two-stage deep learning model for Alzheimer's disease detection and prediction of the mild cognitive impairment time

Shaker El-Sappagh[1,2,6] · Hager Saleh[3] · Farman Ali[4] · Eslam Amer[5] · Tamer Abuhmed[6]

Received: 30 September 2021 / Accepted: 29 March 2022

Abstract
Alzheimer's disease (AD) is an irreversible neurodegenerative disease characterized by thinking, behavioral and memory impairments. Early prediction of conversion from mild cognitive impairment (MCI) to AD is still a challenging task. No study has been able to predict the exact conversion time of MCI patients. In addition, most studies have achieved poor performance making this prediction using only a small number of features (e.g., using only MRI images). Therefore, previous approaches have not gained the trust of medical experts. This study proposes a novel two-stage deep learning AD progression detection framework based on information fusion of several patient longitudinal multivariate modalities, including neuroimaging data, cognitive scores, cerebrospinal fluid biomarkers, neuropsychological battery markers, and demographics. The first stage of the progression detection framework employs a multiclass classification task that predicts a patient's diagnosis (i.e., cognitively normal, MCI, or AD). In the second stage, a regression task that predicts the exact conversion time of MCI patients is used. The study is based on data of 1,371 subjects collected by the Alzheimer's Disease Neuroimaging Initiative (ADNI). Comprehensive experiments were carried out to evaluate the framework stages and find the optimal model for each stage. Proposed model was compared with various machine learning models, including decision tree (DT), random forest (RF), support vector machine (SVM), logistic regression (LR), and K-nearest neighbor (KNN). In the classification stage, the proposed long-short term memory (LSTM) model achieved an accuracy of 93.87%, precision of 94.070%, recall of 94.07%, and F1-score of 94.07%. The results showed that the LSTM model outperformed other machine learning models (i.e., decision tree by 2.48%, random forest by 1.27%, support vector machine by 1.86%, logistic regression by 1.59%, and K-nearest neighbor by 14.77%). In the regression stage, the proposed LSTM model achieved the best results (i.e., mean absolute error of 0.1375). Compared to other regular regressors, this LSTM model achieved less errors (i.e., 0.0064, 0.0152, 0.0338, 0.0118, 0.0198, and 0.0066, compared to DT, RF, SVM, LR, and KNN, respectively). By learning deep representation from patient high-dimensional longitudinal time-series data, the proposed LSTM model was more stable and medically acceptable. The framework may have a clinical impact as a predictive tool for AD progression detection due to its accurate results to predict the exact conversion time of MCI cases using patient time-series multimodalities data.

Keywords Alzheimer's disease · Alzheimer's progression detection · Time-series data analysis · Deep learning

1 Introduction

Alzheimer's disease (AD) is an irreversible and incurable neurodegenerative disease. It is the most common form of dementia in the elderly [1]: 60%–80% of dementia patients have AD [2]. AD has a debilitating effect on patients; it takes a devastating financial toll on society and causes immense challenges for the patients' caregivers. In 2018, the estimated worldwide cost of dementia was US [3] trillion; this is expected to double within 12 years. In 2019, 50 million people had dementia; this number is expected to reach 152 million by 2050 [4]. AD pathology occurs several years before the onset of clinical symptoms, making it difficult to detect the disease in the early stages [5]. MCI is an intermediate stage between being cognitively normal (CN) and having AD. The progression from MCI to AD occurs at an annual rate of 10–25% [6]. MCI has two types: stable MCI (sMCI) and progressive MCI (pMCI). After some time, pMCI patients will convert to AD, but sMCI patients do not convert. Since AD cannot be cured or

Extended author information available on the last page of the article

Published online: 08 May 2022

prevented, early prediction of possible pMCI progression before the occurrence of irreversible brain damage is of tremendous importance to allow preventive care and personalized medicine that led to an improved quality of life.

Most existing studies in the AD domain have either detected AD patients [7] or have predicted their progression after a fixed period of time [8]. These problems are formulated as either binary (e.g., CN vs. AD [9], MCI vs. AD [10] and sMCI vs. pMCI [8, 11]) or multiclass (CN vs. MCI vs. AD [12] and CN vs. sMCI vs. pMCI vs. AD [13, 14]) classification tasks. Few studies have formulated the problem as a regression task [15, 16], and these models fit logistic or polynomial functions to the longitudinal dynamics of each biomarker separately. Moreover, these studies used only magnetic resonance imaging (MRI) data to predict AD progression or to detect the current patient class [13, 17–21]. Rallabandi et al. [13] used a support vector machine (SVM) and MRI measurements of the regional cortical thickness of both left and right hemispheres to detect sMCI, achieving an overall accuracy of 75%. Their study used baseline MRI data only and provided physicians with a suggested class only. Jin et al. [14] collected 16 features from MRI images and used these together with a mini-mental state examination (MMSE) score to predict the pMCI class. They achieved a prediction accuracy of 56.25% using a boosting tree ensemble classifier. The goal of these models was to predict the specific class of the patient, such as CN, MCI, or AD. However, the results obtained in this way are not precise enough. They are insufficient to support medical decision-making because they are based on single modalities and non-time series data and the fact that they predict the patient class only. Moreover, the patient's exact conversion time remains unknown; finding this information deserves more attention.

AD data are multimodal and time series in nature [2, 22–25]. Each patient has a collection of supplemental data of various types, including MRI, positron emission tomography (PET), neuropsychological battery, and cognitive scores (CSs). These heterogeneous modalities carry supplementary knowledge that describes the disease status from different viewpoints [26]. In addition, a patient's data at a certain point in time are not independent of the data at a previous point in time [27]. To predict more precisely, fusing multimodalities and dealing with time-series data are critical design tasks because the resulting feature space is able to provide a holistic picture of a patient's condition [4, 28]. The resulting systems are more accurate and stable and consequently more acceptable from a medical viewpoint [29–31]. A few studies have implemented traditional time series algorithms for the AD progression detection problem [32, 33], but these studies did not investigate the correlation among the patient's multimodal data and how such data evolved [34]. Li et al. [35] asserted the importance of multimodality and longitudinal analysis in AD data. Qiu et al. [36] combined a feed-forward neural network (FFNN) and a VGG-11 NN to study the role of multimodal decision fusion of MRI, MMSE, and logic memory tests to enhance MCI diagnosis. They achieved an accuracy of 90.9%. However, this study was based only on the data of baseline visits and the prediction task formulated as a binary (CN vs. MCI) classification problem. Forouzannezhad et al. [37] used a Gaussian-based model to predict MCI based on demographics, PET, and MRI data and achieved an accuracy of 78.8%.

Over the last decade, conventional (i.e., not deep learning) machine learning (ML) algorithms, such as SVM and random forest (RF), have been applied to MCI conversion prediction [2, 29, 38–40]. Most studies have used single-modality models for addressing binary classification problems such as sMCI vs. pMCI [36, 40]. Zhang et al. [2] provided a survey of pMCI conversion studies. The existing ML studies have depended on a limited number of biomarkers, which may be insufficient to provide a complete interpretation of the disease.

For example, Liu et al. [41] built their model based on MRI and cerebrospinal fluid (CSF) modalities. Lu et al. [6] utilized Fluorodeoxyglucose PET (FDG-PET) imaging data to identify CN subjects who will convert to MCI. They built a binary classifier using an incomplete RF–robust SVM approach and achieved an accuracy of 90.53%. Further, most studies have neglected the temporal dependency within feature series and among different features and focused on cross-sectional data. Cho et al. [42] used the Alzheimer's Disease Neuroimaging Initiative (ADNI) data to predict probable AD conversion using single baseline MRI scans. All in all, we observe that building accurate models based on multimodal time series data with conventional ML classifiers has critical limitations [27].

On the other hand, deep learning (DL) techniques have shown promising results in terms of prediction in several medical areas [43, 44]. Recurrent neural networks (RNNs), extract deep longitudinal features from fused multivariate time series data [21, 34, 45]. For example, Tabarestani et al. [46] used two variations of RNNs, namely long short-term memory (LSTM) networks and gated recurrent units, to predict a patient's status at the following three time-points using the previous three historical time-points. Choi et al. [47] used a convolutional neural network (CNN) to detect pMCI cases based on a single source (i.e., PET images). Spasov et al. [45] proposed a CNN-based multimodal model to detect AD progression, where authors used a late fusion strategy to fuse patients' MRI data, demographic, neuropsychological, and ApoE4 genetic data. Alternatively, Liu et al. [48] proposed a CNN model to jointly predict a patient's diagnosis and a few clinical

scores. The model was based on the fusion of MRI and three demographic features collected from baseline visits only.

As far as we know, no research in ML has been dedicated to predicting the specific time of conversion for MCI patients using regression. In addition, no study has been dedicated to combining AD prediction as a multiclass classification problem with MCI conversion time prediction as a regression task. Formulating AD progression detection as a four-class classification (i.e., CN vs. sMCI vs. pMCI vs. AD) is a complex task that achieves poor results [14]. Relaxing this problem into less complex classification tasks (i.e., a three-class problem) improves the performance of ML algorithms. Gupta et al. [49] converted the four-class task into a series of binary classification tasks comprising AD vs. CN, sMCI vs. pMCI, and so on. However, these models only predict whether the MCI patients will or will not convert to AD within a certain period, such as five years. Unfortunately, these predictions are not helpful to physicians in medical environments. Prediction of the exact conversion time would provide more insightful information about the future status of MCI patients. To the best of our knowledge, no studies have solved this problem. Our study makes the following contributions:

- We propose multistage modeling to integrate the classification and regression tasks for determining whether a patient is MCI and then determining the possible progression time. Namely, we design a two-stage LSTM-based DL model for AD progression detection. In the first stage, a three-class classification model is proposed to detect a patient's class (CN vs. MCI vs. AD), and in the second stage, a regression model is proposed to predict the conversion time for pMCI patients. LSTM has the ability to learn long-term dependencies among time series and within single time series [10], which makes it suitable for learning deep representative features from multivariate time series data.
- The model was implemented based on the early fusion of multimodal time-series data, including neuroimaging data, CSs, CSF biomarkers, neuropsychological battery markers, and demographics. These data were collected from the ADNI dataset, and they are related to four time steps for each patient: baseline (BL), month 6 (M06), month 12 (M12), and month 18 (M18).
- Comprehensive optimization pipelines were implemented based on the data collected from 1,371 subjects in the ADNI. For each pipeline, a set of data preprocessing steps and a separate feature selection and ML algorithms were selected. We explored a set of heterogeneous ML techniques, such as RF, SVM, decision tree (DT), Ridge, Lasso, k-nearest neighbor (KNN), and logistic regression (LR). Also, two DL models were investigated: LSTM and FFNN.
- Time-series data were explored in four different ways: (1) use of the four-time steps data, (2) summarization of time-series data using the average statistics, (3) usage of BL visit data only, and (4) usage of M18 visit data only.
- The designed models show the important role of DL models in extracting deep representative features from time series data in both classification and regression tasks.

The rest of this paper is organized as follows. Section 2 presents our methodology, including a description of the dataset, the ML algorithms used, and the architecture of the proposed framework. Section 3 discusses the experimental setup of the study. Section 4 provides readers with the experimental results. Section 6 concludes the paper.

2 Materials and methods

In this section, we discuss the dataset used, the list of classification and regression algorithms, and the proposed AD diagnosis and MCI time of conversion prediction model.

2.1 ADNI dataset

The dataset contains data from 1,371 subjects (46.5% female) in the ADNI database. As shown in Fig. 1, patients were categorized into four groups based on the individual's clinical diagnosis at baseline and at future time points: (1) CN: 419 subjects diagnosed to be CN at baseline and who remained CN in all future time steps; (2) sMCI: 473 subjects diagnosed to be MCI at all time points; (3) pMCI: 140 subjects evaluated to be MCI at baseline+M06+M12+M18 visits, and who progressed to AD within 2.5 years from M18 (i.e., by the M48 visit); (4) AD: 339 subjects diagnosed as AD in all visits. Subjects who showed improvement in their diagnoses during follow-ups were excluded from the study. For example, those who were clinically diagnosed as MCI but reverted to CN and those who were clinically diagnosed as AD but reverted to MCI or CN were considered misdiagnosed because AD is an irreversible form of dementia. Further, cases that converted directly from CN to AD were also discarded. Several biomarkers and neuropsychological tests were collected and individually validated for AD progression. The following features were considered potential predictors for AD progression, in accordance with results reported in previous studies [1, 12, 21, 29, 35, 50–52]:

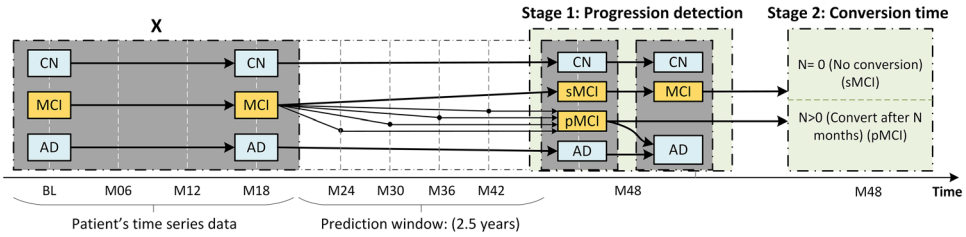

Fig. 1 Format of the patients time series data

- The first group of features corresponds to time series data. These data were collected at baseline and regularly every six months up to month 18 (i.e., over 1.5 years and 4 readings). These data were split into three modalities: (1) CSs (9 features), (2) neuropsychological screening battery (NSB) (51 features), and (3) MRI scans (312 features). CS features including the Alzheimer's disease assessment scale (ADAS 11 and ADAS 13), clinical dementia rating-sum of boxes (CDRSB), global clinical dementia rating (CDGLOBAL), functional assessment questionnaire (FAQ), geriatric depression scale (GDTOTAL), MMSE, Montreal cognitive assessment (MoCA) and neuropsychiatric inventory score (NPISCORE). The NSB features included the Rey auditory verbal learning test (RAVLT) features, daily cognition report, and so on. The imaging data used in our experiments were based on a preprocessed set of T1-weighted MRI features from the ADNI database. Data were pre-processed with the standard ADNI pipeline by a team from the University of California at San Francisco, who performed cortical reconstruction and volumetric segmentation with the FreeSurfer image analysis suite according to the atlas generated in Desikan et al. [53]. The MRI features were associated with regional cortical thickness, regional volume, and surface area measures. Details of the analysis procedure are available online and upon request from the authors. Supplementary file 2 includes the full details about the used features in this study.
- The second group of features corresponds to static baseline data (BL) collected only at the baseline visit. These data included 15 features, such as CSF biomarkers (3 features from the amyloid-$-\beta$ peptide of 42 amino acids, Aβ_{1-42} [Abeta], tau and phosphorylated tau [PTAU]); genetic information (one feature from ApoE4) family history questionnaire (1 feature from mother's dementia history); neuropathology symptoms (4 features from history of low energy, crying, depression, and insomnia); sociodemographic information (4 features from age, body mass index, gender, and years of education); and medical history (1 feature from psychiatric history) and CSF lab test (1 feature from white blood cell count).

The ADNI subjects were aged from 55 to 92 years, were fluent in English or Spanish, and had at least six years of education. At baseline, subjects met the specific inclusion criteria described in Table 1. They met the National Institute of Neurological and Communicative Disorders and Stroke, and the Alzheimer's Disease and Related Disorders Association (NINCDS-ADRDA) diagnostic criteria for probable AD [54]. Supplementary File 1 contains a complete list of the roster ID (RID) used in our study. Detailed descriptions of the ADNI subjects, image acquisition protocol procedures, and post-acquisition pre-processing procedures can be found at http://www.adni-info.org, and upon request from the authors. Demographic and clinical information about the subjects is shown in Table 1.

2.2 Classification models

2.2.1 Decision tree

A DT is a directed acyclic graph where nodes can be decision points (internal nodes) or output points (leaves), and edges connect nodes from the root of the tree to the leaves. This algorithm learns decision rules from training data by recursive partitioning in the input feature space X. Each internal node is associated with a set of records T, which are split based on a specific feature. We implemented the DT classifier and regressor using the classification and regression trees (CART) algorithm. CART creates binary DTs by determining the optimal splitting feature using an appropriate impurity criterion. The classification task uses Gini or entropy impurity, and the regression task uses variance reduction based on the least-squares or mean squared error (MSE). For a dataset $D = \{(X_i, y_i)\}_{i=1}^{N}$, where instance $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ has p features, N is the size of the training set and $y_i \in \{C_1, C_2, \ldots, C_Q\}$ for the multiclass classification task or $y_i \in \mathbb{R}$ for the regression task. If attribute x_{ij} is continuous, it can be split as $x_{ij} \leq$, and if $x_{ij} = \{b_1, b_2, \ldots, b_k\}$ is categorical; then a separate branch is created for $x_{ij} = b_n, n \in 1, 2, \ldots, k$ All candidate splits are generated and evaluated using the splitting criterion. An impurity measure is used to select the best split, and the parent impurity should be decreased by splitting. If $(E_1, E_2, \ldots, E_k)$ is a split of the E

Table 1 Descriptive statistics about the ADNI dataset at baseline

	CN (n = 419)	sMCI (n = 473)	pMCI (n = 140)	AD (n = 339)	Combined (n = 1371)
Gender (M/F)	191/228	283/190	86/54	187/152	747/624
Age (years)	73.84 ± 05.78	72.92 ± 07.76	73.89 ± 06.84	75.01 ± 07.81	73.82 ± 07.18
Education (years)	16.43 ± 02.70	15.80 ± 02.97	16.13 ± 02.71	15.13 ± 02.98	15.85 ± 02.90
FAQ	00.19 ± 00.73	02.10 ± 03.13	04.55 ± 04.54	13.32 ± 06.85	04.54 ± 06.65
MMSE	28.98 ± 01.14	27.63 ± 02.13	26.45 ± 02.09	21.94 ± 03.64	26.59 ± 03.63
MoCA	25.68 ± 01.97	23.14 ± 02.70	21.78 ± 01.99	17.48 ± 03.54	22.38 ± 04.08
ApoE4	00.27 ± 00.48	00.51 ± 00.66	00.85 ± 00.71	00.85 ± 00.71	00.56 ± 00.67
ADAS-Cog 13	08.70 ± 04.09	14.80 ± 05.84	19.62 ± 05.05	30.00 ± 07.99	17.19 ± 10.03
RAVLT immediate	45.80 ± 09.72	36.41 ± 10.65	29.69 ± 07.08	22.64 ± 07.47	35.20 ± 12.80
CDR	00.084 ± 0.30	01.37 ± 00.86	02.07 ± 01.00	05.34 ± 02.21	01.96 ± 02.34

*Data are mean ± standard deviation.

record set, a splitting criterion that makes use of the impurity measure I can be represented as in Eq. 1.

$$\Delta = I(E) - \sum_{i=1}^{k} \frac{|E_i|}{|E|} I(E_i) \tag{1}$$

For the Gini index, let $p_j = \frac{|\{t \in E : t[C] = c_j\}|}{|E|}$ be the probability that a tuple in D belongs to a class c_j, then $\text{Gini}(E) = 1 - \sum_{j=1}^{Q} p_j^2$. Entropy $H(S)$ is the amount of uncertainty in dataset S. Information gain $I(x_i)$ is a measure of the difference in entropy before and after the set S is split based on x_i (see Equation 2).

$$I(S, x_i) = H(S) - \sum_{v \in Value(x_i)} \frac{|S_v|}{|S|} H(S_v) \tag{2}$$

where $\text{Value}(x_i)$ set of all possible values for an attribute x_i, $S_v = \{s \in S \mid x_i(s) = v\}$, and $H(S) = -\sum_{i=1}^{Q} p_i \log_2 p_i$ where Q is the number of classes and p_i is the proportion of S belonging to a class C_i.

For the regression tree, the variance reduction of a node N is the total reduction in the variance of the target variable y because of the node split (see Eq. 3), where S, S_f and S_t are the presplit indices, the set of sample indices for which the split test is false, and the set of sample indices for which the split test is true [55].

$$\begin{aligned} I_V(N) = & \frac{1}{|S|^2} \sum_{i \in S} \sum_{j \in S} \frac{1}{2} (x_i - x_j)^2 \\ & - \left(\frac{1}{|S_f|^2} \sum_{i \in S_f} \sum_{j \in S_f} \frac{1}{2} (x_i - x_j)^2 \right. \\ & \left. + \frac{1}{|S_t|^2} \sum_{i \in S_t} \sum_{j \in S_t} \frac{1}{2} (x_i - x_j)^2 \right) \end{aligned} \tag{3}$$

2.2.2 Random forest

Random forest (RF) [56] is an ensemble classifier formed by a family of T DTs, $h(n_1 \mid X_1), \ldots, h(n_T \mid X_T)$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ is a list of p features for i's DT, and n_i represents the training instances. Each tree T leads to a classifier. Specifically, given data $D = \{(X_i, y_i)\}_{i=1}^{N}$, we train a family of classifiers, h_T. The predictions of all Ts are combined by using the majority-voting mechanism. To create an uncorrelated collection of trees, RF combines bagging and random feature selection techniques. Training set n_i of the i^{th} tree is constructed by bootstrapping n_i examples at random from the N available instances in the whole dataset. RF uses the out-of-bag error estimation to estimate the generalization error of the final model. The bootstrap sampling procedure ensures that approximately one-third of the available N examples are not present in the training set of each tree. For each constructed DT, this third of the original dataset, called the out-of-band (OOB) data, is predicted by the DT, and is consequently used as test data. The averaged prediction error for each training case, x, using only the trees that do not include x in their bootstrap samples, is the OOB error estimate. For each DT, each time a split is considered, a fresh random sample of m predictors (e.g., $m \approx \sqrt{F}$) is chosen as split candidates from the full set of F predictors. The best split is determined and used. The process is repeated until a stopping criterion is met. A node is partitioned using the best possible binary split. Outliers are likely to be ignored by most trees, which makes RF more stable.

2.2.3 Support vector machine

An SVM is a non-probabilistic classifier. It learns the training dataset to find the dividing hyperplane that separates classes with the maximum margin. The predicted

label of a new instance is calculated based on the side of the hyperplane on which it falls. An SVM can be linear or non-linear based on the used kernel function. A linear SVM finds a hyperplane that is a linear function of the input features, as shown in Eq. 4.

$$\min_{w,b,\xi} f(w,b,\xi) = \frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i \tag{4}$$

subject to $y_i(w^T x_i + b) - 1 + \xi_i \geq 0, i = 1,2,\ldots,n, \xi_i \geq 0$ where w denotes the normal vector to the hyperplane, b is the parameter that controls the offset of the hyperplane from the origin along its normal vector, ξ_i is a slack variable to ensure that SVMs can deal with outliers in the data. For each training example x_i, ξ_i gives the distance by which x_i violates the margin in units of $|w|$, and hyperparameter $C > 0$ determines how heavily a violation is punished. Note that here an L_1 regularization term is used. The SVM classifier depends only on a few training points (i.e., support vectors) to classify new instances. The non-linear SVM classification finds a hyperplane that is a non-linear function of the input variable by implicitly mapping an input variable into high-dimensional feature space.

This process is called the kernel trick [57]. SVMs can be used in multiclass classification tasks using many techniques, including one-versus-all SVMs, all-versus-all SVMs [58], and so on. SVMs are one of the most popular algorithms for supervised learning that can effectively deal with high-dimensional datasets. The most important disadvantage of SVMs is that they do not directly provide probability estimates, and thus their decisions are hardly interpretable. SVMs were used for both the classification and the regression tasks. Various implementations are available for the support vector regressor, such as epsilon-support vector regression and nu-support vector regression [59].

2.3 Logistic regression

Logistic regression (LR) [60], also called logit regression, log-linear classifier, or maximum-entropy classifier, is a linear classification model. In this algorithm, the possible outcomes are the probabilities described by the logistic function $|f(x) = 1/(1+e^{-wx})$. This algorithm can fit binary, one-versus-rest, and multiclass classification problems. For multinomial classification, LR uses the SoftMax function to compute $p(y = c \mid x) = \frac{e^{w_c^T x + b_c}}{\sum_{i=1}^{K} e^{w_j^T x + b_j}}$, for $c \in K$ classes. The penalized cost function to be minimized is formulated in the convex optimization Eq. 5, where the LR uses the conditional maximum likelihood for the estimation of parameters w and b.

$$\min_{w,c}\left[\sum_{i=1}^{n}\sum_{c=1}^{K} \mathrm{I}[y_i = c]\log\left(\frac{e^{(w^T X_i + b)}}{\sum_{c=1}^{K} e^{w_c^T x + b_c}}\right) + P\right] \tag{5}$$

where I[.] is the indicator function: $\mathrm{I}[y_i = c] = 1$ if $y_i = c$ is true and 0 otherwise; P is the regularization term, which could be $\frac{\lambda}{2}w^T w$ for ℓ_2 regularization, $\lambda\|w\|_1$ for ℓ_1 for ℓ_1 regularization and $\frac{1-\rho}{w}\ell_2 + \rho\ell_1$ for elastic net regularization; ρ controls the strength of ℓ_1 and ℓ_2 regularization.

2.3.1 K-nearest neighbour

KNN is a non-parametric technique in which the predicted class of a new patient uses the 'feature similarity' method. Given a predefined k, a new instance is classified based on its distance to the $\underline{k}$ training examples. Here, the distance between two instances is measured by the mixed normalized Euclidean distance function (see Eq. 6). For numerical features, the normalized Euclidean distance is calculated, and for categorical features, the distance is 0 if both values are the same and 1 otherwise.

$$\begin{aligned}\mathrm{dist}(X,Y) &= \sqrt{\frac{\sum_{i=1}^{m}(x_i - y_i)^2}{m}}, \mathrm{dist}(X,Y)\\ &= \sum_{i=1}^{m} d(x_i, y_i) \text{ for } d(x_i, y_i)\\ &= \begin{cases} 0 & x_i = y_i \\ 1 & x_i \neq y_i \end{cases}\end{aligned} \tag{6}$$

where X and Y instances are represented by $X = (x_1, x_2, \ldots, x_m)$ and $Y = (y_1, y_2, \ldots, y_m)$ and m is the feature space dimensionality. The class label of the new instance is assigned as the majority of the k-nearest examples, and the predicted value of the regression task is the average value of the k-nearest examples.

2.3.2 Ridge and Lasso regression

Ridge is a regularized linear regression model. The goal is to predict a real value y given a vector X. Its conditional likelihood is formulated as a multivariate normal distribution. The mean parameter is calculated as a linear combination of X $(y \mid X \sim \mathcal{N}(W^T X, \sigma^2 I)$. A maximum posterior estimation convex framework optimizes the weight parameters W according to Eq. 7. The regularization term is used to penalize weights that can never be 0.

$$W_{\min} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2 + \lambda \|w\|_2^2 \tag{7}$$

The Lasso regression model is very similar to the Ridge regression model. Instead of using the ℓ_2 norm, it uses the ℓ_1 norm, as shown in Eq. 8. This model is suitable for modeling high-dimensional data because the regularization term penalizes weights up to 0, which causes some features to disappear and reduces the variance in the model.

$$W_{\min} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2 + \lambda \|w\|_1 \tag{8}$$

2.3.3 Feed-forward neural network

The FFNN, also known as the multilayer perceptron (MLP), is the most common type of neural network in scientific literature. It consists of one input, one or more hidden layers, and one output layer. The input layer receives the input vector, and each input is represented by a neuron. Hidden layers perform the non-linear transformation of the data to extract hidden patterns from input data. The output layer is used to make classifications based on a cross-entropy function or perform a regression based on a linear function. The training process is realized through a backpropagation algorithm that optimizes the cost function based on a specific optimizer. These include optimizers such as Adam, RMSprob, and stochastic gradient descent, among others. The backprop algorithm uses the gradient descendant to derive the squared error concerning the network weight assigned. Let us assume that we have a two-layer FFNN model. For D transformation functions of the form $f(x, w)$, where $f : \varnothing(x) \rightarrow y$ is a combination of a fixed set of linear or non-linear functions of the input variable. Our objective is to learn the weights w of these functions. The classification or regression task is formulated as in Equation 9.

$$f\left(x, w^1, w^{(2)}\right) = \emptyset^{(2)}\left(\emptyset^{(1)}\left(x^T w^{(1)} + b^{(1)}\right)^T w^{(2)} + b^{(2)}\right) \tag{9}$$

where $w^{(1)} = \left(w_1^{(1)}, w_2^{(1)} \ldots w_M^{(1)}\right)^T$, $\emptyset^{(1)} = \left(\emptyset_1^{(1)}, \emptyset_2^{(1)} \ldots \emptyset_D^{(1)}\right)^T$, $w^{(2)} = \left(w_1^{(2)}, w_2^{(2)} \ldots w_D^{(2)}\right)^T$, and $\emptyset^{(2)} = \left(\emptyset_1^{(2)}, \emptyset_2^{(2)} \ldots \emptyset_p^{(2)}\right)^T$ M input variables,$b^{(i)}$ are bias terms, D units in the hidden layer, P units in the output layer. The elements of the input vector x are the units of the input layer, $\emptyset_i^{(1)}$ are the neurons of the hidden layer and $\emptyset_{\mathrm{i}}^{(2)}$ are the neurons of the output layer. Each layer has a non-linearity activation function selected as a hyperparameter in the training process. The activation function in the output layer is selected based on the type of task, for example, SoftMax for classification tasks or a linear function for regression tasks.

2.3.4 Long short-term memory

An LSTM is a kind of recurrent neural network, a subcategory of an artificial neural network. As AD data are time series in nature, RNN models, especially LSTM, are suitable to capture long-term temporal dependencies by solving the exploding and vanishing gradient problem during backpropagation through a time optimization process [61]. In our proposed model, we added LSTM layers to capture temporal patterns from the patient's longitudinal data. The patient's data can be seen as multivariate time series data over four-time steps that we feed to the LSTM block for the sake of capturing the temporal features in the data. As shown in Figure 2, the LSTM cell uses three gates: the input gate (i_{t_n}), forget gate (f_{t_n}) and output gate (o_{t_n}). These gates are responsible for updating, maintaining, and deleting information flow in the cell state. $C_{t_n}, C_{t_{n-1}}$ and $\tilde{C}_{t_n}$) are the cell status at the time t_n, cell status at t_{n-1} , and the updated cell status at t_n, respectively. $h_{t_{n-1}}$ is the output value from each memory cell in the hidden layer at the previous time step. h_{t_n} is the hidden layer's value at time t_n based on $\tilde{C}_{t_n}$) and $C_{t_{n-1}}$ · θs are the set of weight matrices, and bs are the biases vectors, which are updated using the backpropagation algorithm. Besides, $\otimes$ represents the Hadamard product; σ is the standard logistic sigmoid function; $\oplus$ is the concatenation operator; and φ is the output activation function, for example, SoftMax or Tanh. Equations 10, 11, 12, 13,14, 15 and 1616 illustrate the information flow in the memory cell at a given time step.

$$f_{t_n} = \sigma\left(\theta_f \cdot [h_{t_{n-1}}, x_{t_n}] + b_f\right) \tag{10}$$

$$i_{t_n} = \sigma(\theta_i \cdot [h_{t_{n-1}}, x_{t_n}] + b_i \tag{11}$$

$$\tilde{C}_{t_n} = \tanh(\theta_C \cdot [h_{t_{n-1}}, x_{t_n}] + b_C) \tag{12}$$

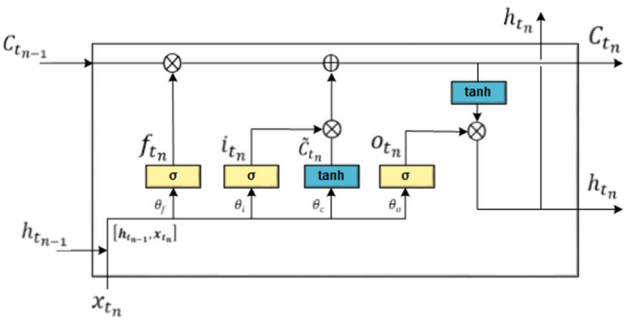

Fig. 2 LSTM cell

$$C_{t_n} = \left(f_{t_n} \otimes C_{t_{n-1}} \oplus i_{t_n} \otimes \tilde{C}_{t_n}\right) \quad (13)$$

$$o_{t_n} = \sigma(\theta_o \cdot [h_{t_{n-1}}, x_{t_n}] + b_o) \quad (14)$$

$$| \; h_{t_n} = o_{t_n} \otimes \tanh(C_{t_n}) \quad (15)$$

$$y_n = \varphi\left(\theta_y h_{t_n} + b_y\right) \quad (16)$$

2.4 Proposed framework

This study proposes a two-stage framework to detect the progression of specific patients (a classification task). Once the patient progression is detected in the future (M48) using the patient's time-series multimodal data (BL+M6+M12+M18), the second stage is designed to predict the conversion time (a regression task) for the progressed patients who are classified in the past as MCI class. As shown in Fig. 1, the pMCI patients were merged at M48 with the AD patients to build the progression detection model for the first stage. The second stage is designed to collect the MCI patients' (i.e., sMCI and pMCI) time-series data and build a regression model that predicts the patient's exact conversion month. Therefore, the main goal of the first stage is to use the patient's time-series multimodal data to predict the diagnosis after 2.5 years (M48). If a progression is detected, the second stage determines the exact conversion month the pMCI patient. The two tasks are trained and tuned independently based on multimodal time-series data. For the classification task, we trained various popular ML models, including DT, RF, SVM, LR, and DL models, including FFNN and the LSTM architecture. The same process was carried out for the regression task using ML regressors, including also conventional ML models (DT, RF, SVR, linear ridge, FFNN, and Lasso) and DL models (LSTM). Consider having M modalities of data represented as $X = \{X^{(1)}, \ldots, X^{(M)}\}$. Each modality X^m is represented as $X^m = \left\{x_1^{(m)}, \ldots x_i^{(m)}, \ldots, x_N^{(m)}\right\}$ for N patients, where each patient $x_i^{(m)} \in \mathbb{R}^{t \times f}$ is a multivariate time series , $x_i = \left\{x_i^{(1)}, \ldots x_i^{(m)}, \ldots, x_i^{(M)}, y_i\right\}$, for $t = 1, \ldots, s$ time steps and the set f of univariate time series. For N patients, each patient i is represented as $x_i = \left\{x_i^{(1)}, \ldots x_i^{(m)}, \ldots, x_i^{(M)}, y_i\right\}$, $i = 1, \ldots, N$, and $y_i \in \{CN, MCI, AD\}$ for the multiclass classification task or $y_i \in \mathbb{R}$ for the regression task. The time-series data were used to train classifiers and regressors using different formatting methods: (1) The multivariate time series data were used with the LSTM model directly, where $s = 4$ for BL, M06, M12, and M18. (2) We collected aggregated features of the four patients' historical visits. We extracted the statistical measurement that described the time series characteristics. For patient P_i, the aggregated feature space was $\mathbb{R}^{M \times f}$, where the collected statistic $\bar{f}$ for each time series statistic $f_{t_1,t_2,\ldots,t_s}$ was the mean $\left(\frac{1}{s}\sum_{i=0}^{S} f_i\right)$ for s time steps, (3) Inspired by the image processing domain, we flattened the time series data of each patient to be represented as a single vector. For patient $P_i \in \sum_{i \in M} X^{(i)}, i = 1, 2, \ldots, N$, and $X^{(i)} \in \mathbb{R}^{t \times f}$, the new representation was $X^{(i)} \in \mathbb{R}^{t \times f}$. Figure 3 represents the flattening of time series data and its usage in an FFNN for the classification task. We tested all these formulations with different representation was $P_i \in \mathbb{R}^{t \times f \times M}$. Figure 3 represents the flattening of time series data and its usage in an FFNN for the classification task. We tested all these formulations with different ML models in both classification and regression tasks. The optimization cost function depends on the technique used. The regularized cross-entropy loss function based on SoftMax was used for the multiclass optimization problem (see Eq. 17). The regression task was tuned using the regularized MSE function as shown in Eq. 18.

$$\min_{\theta} \mathcal{L}(\theta) = -\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{k}\sum_{k=1}^{K} I(y_k = k) \log\left(\frac{\exp\left(\theta^T x_i + b\right)}{\sum_{j=1}^{K}\exp\left(\theta_j^T x_i + b\right)}\right)\right] \quad (17)$$

$$\min_{\theta} \mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \frac{\lambda}{N}\sum_{j=1}^{N}\theta_j^2 \quad (18)$$

where θ represents the network weights parameters, the last term in both equations is the regularization term, y_i and $\hat{y}_i$ i are the actual and predicted values, $I(.)$ is an indicator function, where I(true statement)=1 and 0 otherwise, and the label y can take on K different values, $y_k \in \{1, 2, \ldots, K\}$, and in our case $K = 3$. For each input x, the model calculates the probability that $P(y_k = K \mid x; \theta)$ for each $k \in \{1, \ldots, K\}$. The output is a K-dimensional vector of K estimated probabilities, whose sum is 1. Figure 4 illustrates the development steps to produce our model. We sequentially applied the following steps:

- First, in accordance with the AD literature, a set of three of the most popular time series modalities (CSs, MRI, and NSB) and one combined baseline modality (BL) were collected. This study is based on the early fusion of the three modalities. This fusion method has many advantages over decision and intermediate fusion strategies [25].
- Second, these data were randomly, and using stratified sampling, split into 90% (1233 patients) as model development sets (MDSs) and 10% (138 cases) as model test sets (MTSs). This data splitting process was

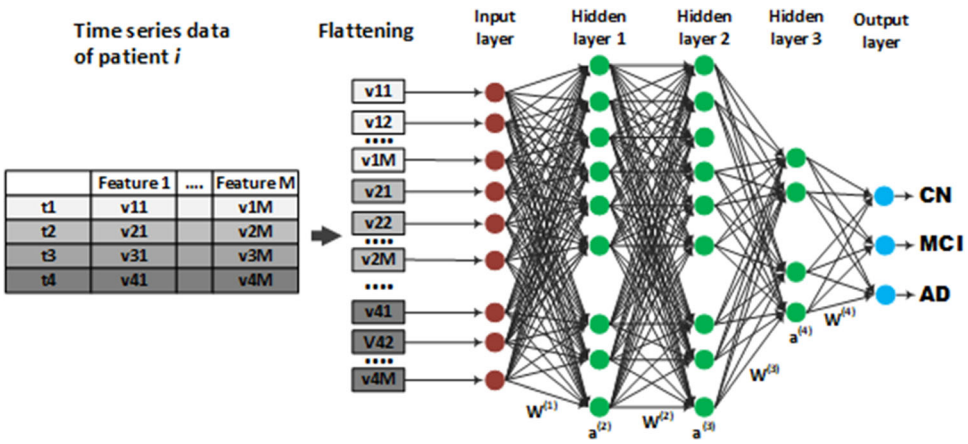

Fig. 3 Time series data flattening process for the FFNN model

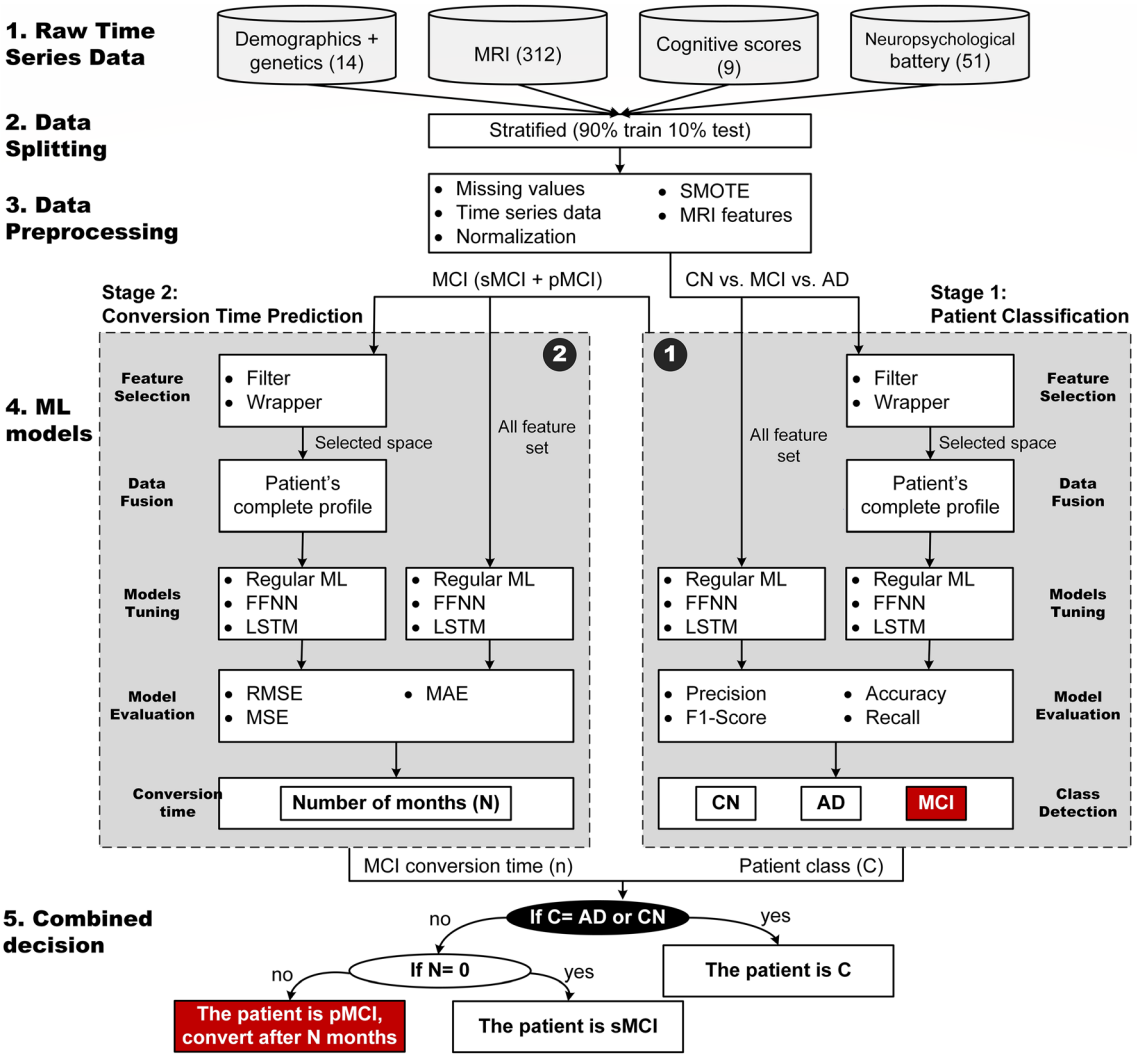

Fig. 4 Proposed two stages deep learning model for hybrid AD diagnosis and conversion prediction

repeated 10 times, and the average performance was recorded. For each train/test split, the MDS was used with the stratified 10-fold cross-validation (CV) technique to optimize ML models and measure the CV performance. This stratified 10-fold CV process is repeated 10 times for every outer train/test split, and the

average performance is then collected. We trained the tuned model using the whole MDS set, and then we used the test set only to evaluate the performance of the final model. This strategy is accurate and not biased because, from the very beginning, we randomly (and in a stratified way) isolated a separate test set to be used for measuring the generalization performance of the ML models. As such, the test set is not used either in data normalization or in model optimization.

- Third, only the MDS was pre-processed based on a pipeline of a set of steps. The pipeline includes handling missing values, time series feature extraction for conventional ML models, data balancing using SMOTE's oversampling technique [62], data normalization, and MRI feature reduction. Two types of data resulting from this step: time-series data to be fed to the LSTM model and aggregated data to be fed to the conventional ML and FFNN. The aggregation of time series data was based on the average value for each time series. Note that, for every patient, his or her BL data were repeated with every time step to be combined with the time-series data. As the MRI modality has a very large set of features, we concentrated our study on the role of some brain regions of interest. The regions used were chosen based on the AD progression detection literature. Recent studies have asserted that AD progression is tightly correlated with atrophies in the structures of the medial temporal lobe (MTL) [63–65]. The MTL includes a set of anatomical regions, such as the hippocampus, amygdala, entorhinal and parahippocampal cortices. Each of these regions has a left and right part. In addition, cortical thickness has a high predictive value for AD progression detection [20][65]. We collected the volume, surface area, and cortical thickness from each of these parts. Of the 312 MRI features, we selected 58 features for volume and cortical thickness of the left and right structures, including HIPPOCAMPUS, AMYGDALA, PARAHIPPOCAMPAL, ENTORHINAL, VENTRICLES, FUSIFORM, INFERIOR/ MIDDLE/ SUPERIOR TEMPORAL, and INSULA.
- Fourth, the resulting data were used to train ML models in parallel over two different stages. The first stage was the patient classification stage, which determined the patient's class (e.g., CN, MCI, or AD). This was a multiclass classification task in which the data from 1,371 patients (1,233 cases for training and 138 cases for testing) were used to train and test the model. The model training was based on the stratified 10-fold CV technique to avoid overfitting. The second stage predicted the exact conversion time of MCI patients as a regression task. The data of 613 MCI cases (both sMCI and pMCI) were used to train and test the models. We used the 10-fold CV technique to validate the regression models. Selecting relevant features from the raw feature space is expected to minimize redundancy, avoid overfitting, improve model generality, and reduce the model's computational cost. There are two main feature selection techniques: filter and wrapper methods. The filter method uses a univariate filtering method such as correlation or information gain to filter out the least promising features. In contrast, wrapper methods are used to measure the importance of specific feature sets. This is based on training and testing a specific classifier using different subsets of features in the space of feature subsets. Each method has its own advantages and disadvantages. In our study, the MRI modality has been used by both the filter and wrapper methods. The same process was followed for the NSB modality. The resulting feature set from both the MRI and the NSB were fused with the list of CS and BL data. The resulting feature vector was used to train different ML and DL models. The same process was followed in both the classification and the regression tasks. For the filter method, we used the information gain technique to assess the dependency of the independent variable in predicting the target variable. For the wrapper method, we used recursive feature elimination with the cross-validation method (RFECV) coupled with RF (either classifier or regressor) for measuring feature importance.
- Fifth, the prepared datasets were used to train and optimize different ML and DL models. In the first stage, we tuned a set of different popular ML classifiers, including RF, SVM, DT, LR, and KNN. For better handling of time series data, we tuned an LSTM-based DL model. The conventional ML models were tuned using a grid search, and the LSTM model architecture and hyperparameters were tuned using the Keras Tuner . In the second stage, we tuned five popular ML regressors: DT, linear ridge, Lasso, RF, and SVM. In addition, we built and tuned FFNN and LSTM models using Keras Tuner. The trained models at both stages were tested using unseen test data to measure the generalization performance of each model. The first stage predicted the patient class as either CN, MCI, or AD. If the patient was CN or AD, the process was finished. If the patient was MCI, then the second stage was called on to predict the exact time when the patient would convert to AD. The output of the second stage was a number n. If n=0 then the patient was sMCI. If n>0, then the patient was pMCI and would convert after n months from the diagnosis time.

2.5 Data pre-processing step

In this section, we improve the quality of our dataset by using three preprocessing steps. These steps are discussed in details in the following subsections.

2.5.1 Handling missing data and data balancing

We first excluded features from the baseline static data if these features have more than 30% of their data points missing. Following that, a KNN-based imputation was applied for the missing real values, where all missing values were filled by average values of clusters formed by subjects with the same diagnosis. In our study, k was set to 10 empirically. Unlike real value imputation, the Euclidean distance was used for numerical value imputation. For categorical values, a distance of 0 was set if both values were the same; otherwise, a distance of 1 was set. The same procedure was applied for time-series data, where we also removed all features with more than 30% missing data and removed all patients with missing baseline readings. For processing the non-existing time series data due to the ADNI procedure, we followed two strategies based on an intuition of the ADNI. First, we filled non-applicable values for every category of data, according to ADNI procedures. For example, if an ADNI patient who was CN has no MRI scan at visit M18, we considered these types of values to be missing, i.e., missing not at random. Also, several lab tests, cognitive tests, and neuroimaging scans were not applicable for specific diagnoses at specific visits. Therefore, we applied an accurate filling procedure, where a forward filling is used when the patient's diagnosis persists. Otherwise, we considered the value missing. This forward filling technique is common in Alzheimer's studies [48]. The second step was to determine missing values from existing data using statistical or ML techniques. We opted to apply a medically intuitive and well-known method. Thus, in the case of numerical data, we used the mean value according to the different classes: CN, sMCI, pMCI, and AD. For categorical features, we used the mode value according to the patient class. We used SMOTE oversampling technique to balance the dataset are remove the possibility for biased predictions.

2.5.2 Data standardization

To guarantee fast convergence of our model as well as ensure that all used features have the same level of importance, all features were standardized using the z-score method, that is, $z_j = \left(x_j - \mu_j\right)/\sigma_j$ where x_j is the original value for feature j, Z_j is the normalized value, μ_j is the feature's mean and σ_j is the feature's standard deviation. As a result, the z-score method produces a new dataset where all features have a zero mean and unit standard deviation. The values of categorical features were also encoded.

3 Experiments setup

To evaluate the performance and effectiveness of our proposed method, we tuned, tested, and compared many machine learning models with different pipeline settings, including the usage of different feature selection techniques, conventional ML algorithms, neural network architectures, and types of data (i.e., time series and non-time series). Inspired by [66], for time series data, a total of four time steps–baseline data (BL), M06, M12, and M18-were used to predict AD at M48. Time series data were used to train an LSTM-based DL model to check the role of these data in increasing the confidence and accuracy of the model. We implemented and tested a set of conventional ML models using both aggregated and flattened time series data. In addition, an FFNN neural network model was tuned using both aggregated and flattened time series data. In addition, we built an FFNN architecture based on the BL and M18 data to check the effect of the time gaps between the observed data and the monthly prediction.

3.1 Performance evaluation

The first stage was a classification task. Four standard metrics were used to evaluate the classification models, i.e., accuracy, precision, recall, and F1-score, where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives (see Eqs. 19, 20, 21 and 22):

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{19}$$

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{20}$$

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{22}$$

The second stage was a regression task. Three metrics were used to evaluate the regressor models: the mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) (see Eqs. 23, 24 and 25), where $\mathrm{N}, y_i, \hat{y}_i$ denote the total number of observations, actual value, and predicted value, respectively:

$$MAE = \frac{1}{N}\sum_{1}^{N}|y_i - \hat{y_i}| \tag{23}$$

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(y_i - \hat{y_i})^2 \tag{24}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_i - \hat{y_i})^2} \tag{25}$$

3.2 ML models training

For all the experiments in this paper, we employed the Python 3.7.3 distributed in Anaconda 4.7.7 (64-bit). The 1371 case dataset was divided in a stratified way into a training set (90%) and testing set (10%), keeping the same percentage of each class in both sets. After a set of preprocessing steps, we tuned a set of different pipelines to explore the role of time series and aggregated data, the LSTM model, the FFNN model, conventional ML algorithms, feature selection paradigms, and hyperparameter tuning. The same pipeline designs were tested over the two stages regarding first classification and then regression tasks. The training data were used for feature selection and hyperparameter optimization based on the stratified 10-fold CV. The selected features were masked on the testing datasets, and the resulting sets were normalized and used to measure the generalization performance of different models. We used a set of standard and popular evaluation metrics to measure the classification and regression models, as discussed in the previous section. To prevent bias, the procedure was repeated 10 times in our experiments. We compared the four-class (CN, sMCI, pMCI, and AD) and three-class (CN, MCI, and AD) classification problems in the first stage. As it has been well established in the literature [14], the four-class task is very challenging. This fact is reinforced in the results section. Given our two-stage design, we depended on the initial three-class classification task because in the second separate stage, not only was the MCI class separated into sMCI and pMCI, but the exact conversion time was also precisely predicted. In both stages, two different groups of data were used. First, the three time-series modalities and BL modality were combined. In this case, each patient will have four time steps, and the BL data are repeated four times for every patient. Second, an aggregated version of the three time-series modalities and BL modality were combined. In this case, the time series data are aggregated, and a single row is used for every patient.

In both stages, the MRI (58 features) and NSB (51 features) modalities were separately fed into the information gain-based feature selection technique, which ranked the features according to their importance levels. We selected the top 29 features from the MRI features and the top 26 features from the NSB features because they achieved the best results. The same modalities were separately fed into the RFECV, and we again selected the top half of features (29 features) from the MRI features and the top half of features (26 features) from the NSB features. Note that these processes were carried out for both the four-class and the three-class problems separately. Selected features from both the MRI and NSB data were fused with the BL and CSs feature sets. The resulting different feature spaces from the filter and wrapper methods had 78 features. These data were used to train the selected ML algorithms in both stages. Each ML model was tuned and tested using the full feature set, the selected features from the filter method, and the selected features from the wrapper method.

In the first stage, grid search was used to optimize the hyperparameters of all the conventional ML models, while Keras Tuner was used to select the best architectures for both the FFNN and LSTM models. These models were tuned for the four-class and three-class classification problems. The same procedure was followed for the second stage, but the models were tuned for regression tasks. Finally, both stages were combined to make the final progression detection prediction. Regarding the LSTM classification models, using Keras Tuner, we tuned three models to learn the complete feature set. In all models, the learning rate was fixed at 0.0001, and the Adam optimizer was used. All hidden layers used the rectified linear unit (ReLU) activation function, and the output layer used the SoftMax activation function. A dropout layer was used after each hidden layer, and L2 regularization was added to prevent overfitting. The training data were shuffled for each training epoch to ensure the optimization was stochastic and to avoid convergence to a local minimum. The batch size was 50, and the number of epochs was 70.

- For the three-class models, we tuned three LSTM architectures. First, the full feature set-based LSTM model had three LSTM layers with 330, 450, and 430 units. The output layer has 3 units (one per class). Except for the output layer, each layer underwent L2 regularization with parameters 0.2, 0.3, and 0.01, respectively, and each layer was followed by a dropout layer with probabilities of 0.3, 0.4, and 0.4, respectively. Second, the wrapper feature set-based LSTM had four LSTM layers with 470, 310, 170, and 3 units. Except for the output layer, each layer underwent L2 regularization with parameters 0.3, 0.4, and 0.1, respectively, and each layer was followed by a dropout layer with probabilities of 0.3, 0.2, and 0.3, respectively. Third, the filter feature set-based LSTM had four LSTM layers with 70, 270, 70, and 3 units. Each layer

underwent L2 regularization with parameters 0.01, 0.5, and 0.01, respectively. Each layer was followed by a dropout layer with probabilities of 0.4, 0.3, and 0.3, respectively.

- For the four-class models, we tuned three LSTM architectures. First, the full feature set-based LSTM had four LSTM layers with 210, 470, 70, and 4 units. Except for the output layer, each layer underwent L2 regularization with parameters 0.05, 0.3, and 0.1, respectively, and each layer was followed by a dropout layer with probabilities of 0.5, 0.3, and 0.1, respectively. Second, the wrapper feature set-based LSTM had four LSTM layers with 470, 310, 370, and 4 units. Except for the output layer, each layer underwent L2 regularization with parameters 0.3, 0.4, and 0.1, respectively, and each layer was followed by a dropout layer with probabilities of 0.05, 0.3, and 0.4, respectively. Third, the filter feature set-based LSTM had four LSTM layers with 490, 350, 410, and 4 units. Each layer underwent L2 regularization with parameters 0.02, 0.01, and 0.1, respectively. Each layer was followed by a dropout layer with probabilities of 0.4, 0.2, and 0.4, respectively.

In the second stage, three LSTM models were trained based on the three feature sets. All hidden layers used the ReLU activation function, and the output layer used the linear activation function. The number of epochs and batch size were the same as in the first stage. First, the full feature set-based LSTM had one LSTM layer with 440 units followed by three dense layers: 60, 80, and 70 units. Except for the output layer (1 unit), each layer underwent L2 regularization with parameters 0.1, 0.01, 0.01, and 0.1, respectively, and each layer was followed by a dropout layer with probabilities of 0.2, 0.3, 0.3, and 0.3, respectively. Second, the filter feature set-based LSTM had one LSTM layer with 440 units followed by four dense layers: 90, 60, 70, and 1 units. Except for the output layer, each layer underwent L2 regularization with parameters 0.1, 0.01, 0.01, and 0.01, respectively, and each layer was followed by a dropout layer with probabilities of 0.4, 0.3, 0.5, and 0.4, respectively. Third, the wrapper feature set-based LSTM had one LSTM layer with 680 units followed by three dense layers: 70, 70, and 1 units. Except for the output layer, each layer underwent L2 regularization with parameters 0.01, 0.05, and 0.01, respectively, and each layer was followed by a dropout layer with probabilities of 0.4, 0.5, and 0.3, respectively.

4 Results and discussion

This section has two subsections. In Sect. 4.1, we discuss the results of the first stage. We compare the ML models in the four-class task and in the three-class task. Then, we compare the two problems. In Sect. 4.2, we discuss the results of the second stage. We compare the performance using default parameters, tuned models, and different architectures for the FFNN design.

4.1 Results of the Patient Classification stage

Each conventional ML model (DT, RF, SVM, LR, and KNN) and LSTM model were tuned based on the full feature set and on the sets selected by the wrapper and filter methods. Tables 2 and 3 show the performance of 18 ML models using four-class classification task. In case of testing, the measured accuracy and F1- score of DT model was 82.82% and 78.80%, respectively, which is surprising higher than other models. In case of CV, the LSTM model registered higher accuracy (84.44%) and F1- score (84.16%) compared to other models. For four-class classification task, the achieved results indicate that the performance of DT and LSTM model in testing approach is higher than SVM, LR, and KNN models. However, the performance of LSTM model in CV method is highest among trained classifiers. Most models achieved their best results based on the filter-based feature set. No model achieved its best performance using the full feature set except for the KNN model. One possible reason for this is that KNN is a lazy learner that memorizes the training set. The KNN model has no learning process. Therefore, KNN achieved the lowest results overall.

Tabels 4 and 5 present the obtained results of all ML models for the three-class classification task using testing and CV methods. The evaluation matrics of ML models were individually identified using both testing and CV methods. In case of testing method, we can see in Table 5 that the obtained accuracy and F1-score of RF were 92.01 and 92.07%, respectively, which is higher than the accuracy (91.22%) and F1- score (91.28%) of LSTM model. However, in case of CV, the performance of all ML models were lower than LSTM model.

The best test performance for all models was achieved by means of the filter feature set except for RF, where wrapper features achieved the best test performance. No model achieved its best performance using the full feature set. These results highlight the role of the feature selection step in optimizing the performance of ML models. Figure 5 illustrates a comparison between the LSTM-based CV and test performance for both four-class and three-class problems. As can be seen, the three-class models achieved more

Table 2 The cross-validation performance of ML models for the four-class task

Models	Dataset	Cross-validation performance			
		Accuracy	Precision	Recall	F1-score
DT	Full feature set	82.89 ± 2.80	78.79 ± 2.00	75.94 ± 2.20	82.89 ± 2.80
	Wrapper method	82.95 ± 2.10	78.86 ± 2.00	76.02 ± 1.90	82.95 ± 2.10
	Filter method	82.95 ± 2.19	78.86 ± 2.04	76.02 ± 1.90	82.95 ± 2.19
RF	Full feature set	83.79 ± 2.30	81.80 ± 2.70	81.27 ± 3.90	83.47 ± 2.50
	Wrapper method	84.16 ± 2.34	81.93 ± 2.46	82.09 ± 3.75	83.74 ± 2.72
	Filter method	84.27 ± 2.60	82.49 ± 2.60	82.65 ± 3.30	84.12 ± 2.50
SVM	Full feature set	81.96 ± 2.80	81.24 ± 2.80	81.13 ± 2.80	81.96 ± 2.80
	Wrapper method	81.95 ± 2.70	81.09 ± 2.60	81.07 ± 2.80	81.95 ± 2.79
	Filter method	81.34 ± 3.03	80.61 ± 2.90	80.46 ± 3.10	81.34 ± 3.00
LR	Full feature set	82.32 ± 3.02	81.24 ± 3.04	81.24 ± 3.48	82.32 ± 3.02
	Wrapper method	82.52 ± 2.84	81.01 ± 2.80	80.99 ± 3.42	82.52 ± 2.80
	Filter method	82.06 ± 2.94	80.99± 2.72	80.83 ± 2.93	82.06 ± 2.94
KNN	Full feature set	70.15 ± 2.97	67.10 ± 3.02	66.74 ± 4.28	70.15 ± 2.97
	Wrapper method	70.88 ± 2.71	67.15 ± 2.70	65.55 ± 4.32	70.88 ± 2.71
	Filter method	70.60 ± 2.60	67.41 ± 2.63	66.95 ± 4.34	70.60 ± 2.60
LSTM	Full feature set	83.43 ± 1.92	83.41 ± 1.68	87.80 ± 2.16	79.55 ± 2.00
	Wrapper method	**84.44 ± 2.20**	**84.16 ± 1.83**	**87.79 ± 1.99**	**80.90 ± 2.26**
	Filter method	83.49 ± 2.55	83.48 ± 2.35	87.43 ± 2.50	79.96 ± 2.68

Table 3 The testing performance of ML models for the four-class task

Models	Dataset	Testing performance			
		Accuracy	Precision	Recall	F1-score
DT	Full feature set	82.17	78.24	76.06	82.17
	Wrapper method	82.75	78.74	76.39	82.75
	Filter method	82.82	78.80	76.42	82.82
RF	Full feature set	80.80	77.08	74.13	80.80
	Wrapper method	80.94	77.42	74.90	80.94
	Filter method	81.23	77.70	75.23	81.23
SVM	Full feature set	71.09	69.52	68.54	71.09
	Wrapper method	71.38	70.40	69.80	71.38
	Filter method	71.45	70.99	70.87	71.45
LR	Full feature set	68.19	66.70	65.96	68.19
	Wrapper method	70.07	68.45	67.59	70.07
	Filter method	72.54	71.63	71.01	72.54
	Full feature set	59.64	55.44	53.35	59.64
	Wrapper method	64.20	59.52	56.90	64.20
	Filter method	59.35	54.75	51.94	59.35
LSTM	Full feature set	77.75	76.82	76.80	77.75
	Wrapper method	**82.80**	**78.74**	**76.39**	**82.80**
	Filter method	79.56	79.32	79.92	79.56

accurate results with every feature set. As a result, the first stage was based on the three-class LSTM classification model. This model determines whether a patient is CN, MCI, or AD. For MCI patients, the second stage's regression model is applied to determine whether the patient is sMCI or pMCI. In the case of pMCI patients, our model determines their exact conversion time.

Table 4 The cross-validation performance of ML models for the three-class task

Models	Dataset	Cross-validation performance			
		Accuracy	Precision	Recall	F1-score
DT	Full feature set	91.39 ± 2.17	91.37 ± 2.20	91.63 ± 2.11	91.40 ± 2.17
	Wrapper method	91.39 ± 2.18	91.36 ± 2.20	91.63 ± 2.11	91.40 ± 2.17
	Filter method	91.40 ± 2.17	91.37 ± 2.20	91.63 ± 2.12	91.40 ± 2.17
RF	Full feature set	92.52 ± 1.89	92.53 ± 2.08	92.50 ± 1.78	92.51 ± 2.09
	Wrapper method	92.52 ± 2.12	92.52 ± 1.99	92.78 ± 2.12	92.52 ± 1.96
	Filter method	**92.60 ± 1.96**	**92.66 ± 2.09**	**92.70 ± 1.94**	**92.62 ± 2.04**
SVM	Full feature set	91.58 ± 2.40	91.57 ± 2.41	91.75 ± 2.39	91.58 ± 2.40
	Wrapper method	92.01 ± 1.97	92.00 ± 1.97	92.16 ± 1.96	92.01 ± 1.97
	Filter method	91.92 ± 2.09	91.91 ± 2.09	92.08 ± 2.08	91.92 ± 2.09
LR	Full feature set	91.71 ± 2.16	91.71 ± 2.16	91.90 ± 2.14	91.71 ± 2.16
	Wrapper method	92.28 ± 2.12	92.28 ± 2.12	92.46 ± 2.10	92.28 ± 2.12
	Filter method	91.89 ± 2.29	91.89 ± 2.29	92.06 ± 2.26	91.89 ± 2.29
KNN	Full feature set	78.95 ± 3.28	78.96 ± 3.28	79.76 ± 3.14	78.95 ± 3.28
	Wrapper method	79.10 ± 3.43	79.14 ± 3.41	80.17 ± 3.20	79.10 ± 3.43
	Filter method	78.95 ± 3.28	78.96 ± 3.28	79.76 ± 3.14	78.95 ± 3.28
LSTM	Full feature set	91.69 ± 3.17	91.92 ± 3.19	91.92 ± 3.19	91.91 ± 3.19
	Wrapper method	**93.87 ± 1.48**	**94.07 ± 1.58**	**94.07 ± 1.58**	**94.07 ± 1.58**
	Filter method	92.73 ± 1.15	93.05 ± 1.24	93.05 ± 1.24	93.04 ± 1.24

Table 5 The testing performance of ML models for the three-class task

Models	Dataset	Testing performance			
		Accuracy	Precision	Recall	F1-score
DT	Full feature set	91.02	90.96	91.57	91.02
	Wrapper method	91.96	91.89	92.54	91.96
	Filter method	91.52	91.46	92.06	91.52
RF	Full feature set	90.29	90.25	90.42	90.29
	Wrapper method	**92.01**	**92.07**	**92.02**	**92.01**
	Filter method	91.67	91.63	91.74	91.67
SVM	Full feature set	76.38	76.37	76.48	76.38
	Wrapper method	79.49	79.50	79.58	79.49
	Filter method	82.82	82.87	83.01	82.82
LR	Full feature set	76.38	76.33	76.35	76.38
	Wrapper method	81.09	81.06	81.10	81.09
	Filter method	81.88	81.82	81.90	81.88
KNN	Full feature set	68.91	68.11	70.29	68.91
	Wrapper method	72.54	71.96	75.35	72.54
	Filter method	68.91	68.11	70.29	68.91
LSTM	Full feature set	88.33	88.45	89.38	88.33
	Wrapper method	**91.22**	**91.28**	**91.84**	**91.22**
	Filter method	89.56	89.71	90.71	89.56

4.2 Results of conversion time detection stage

In this stage, we tuned regression models to predict the conversion time of MCI patients. We tuned five popular ML regressors: DT, Ridge, Lasso, RF, and SVM. In addition, we tuned two neural network architectures, FFNN and LSTM models, using the Keras Tuner package. Results are discussed in the text in terms of MAE because the other metrics of MSE and RMSE are consistent with MAE. Note that the results of all three metrics were reported in the tables. The three metrics are reported in the tables.

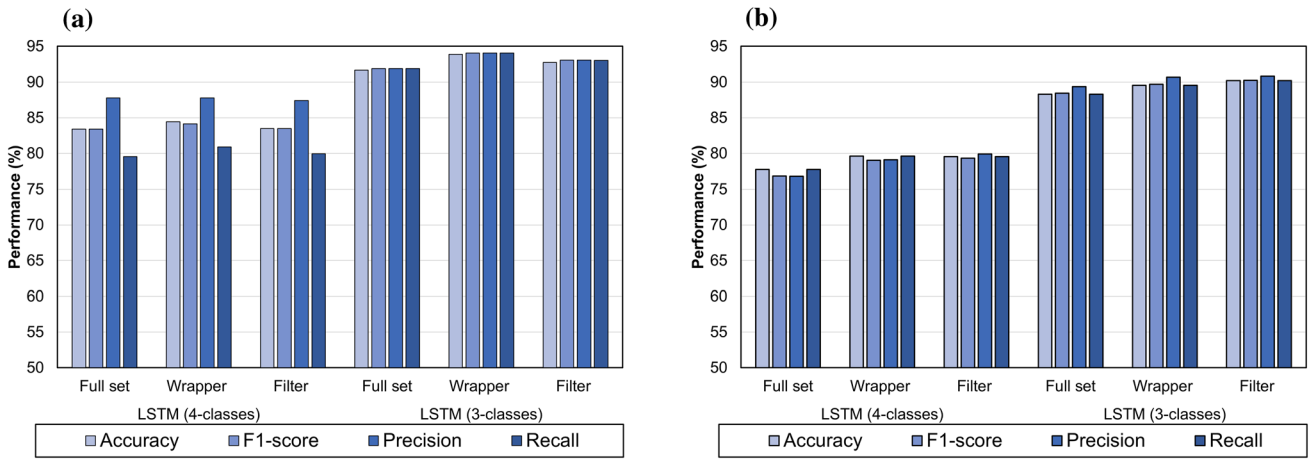

Fig. 5 Comparison of performance LSTM models for three-class and four-class problems, **(a)** CV and **(b)** testing

- In the first experiment, we trained 15 conventional ML models utilizing the default Scikit-learn hyperparameter settings and averaged time-series data fused with BL modality (see Table 6). A separate model was tuned for each algorithm for the complete feature set, wrapper-based feature set, and filter-based feature set. For the DT model, the best MAE was 0.1774 based on wrapper features. Compared with the DT model, the Ridge model achieved better performance of 0.1609 for MAE based on wrapper features, respectively. The Lasso model achieved equal performance using the complete and wrapper feature sets (i.e., MAE = 0.1713). Based on the wrapper feature set, RF achieved better performance than the previous models: MAE = 0.1555. Finally, the best performance was achieved by the SVM model using the wrapper feature set (i.e., MAE = 0.1492). It can be seen that all algorithms achieved better results using the feature selection methods. Most algorithms achieved the best performance based on the wrapper feature sets.
- In the second experiment, we tuned the five conventional ML models-DT, Ridge, Lasso, RF, and SVM–by means of a grid search technique, see Table 7. These models were tuned based on the averaged time series modalities combined with the BL modality. Further, we tuned the LSTM model architecture using the Keras Tuner package. The LSTM model was tuned using the time series modalities combined with the repeated BL modality. Finally, we built an tuned FFNN architecture based on the Keras Tuner. The model was trained using the averaged time series data fused with the BL modality. As can be clearly seen, the LSTM model achieved the optimum MAE of 0.1375. The tuned DT model achieved the second-best results of MAE = 0.1439 based on the filter feature set. The FFNN model achieved the third-best results of MAE = 0.1441 based on the filter feature set. The best results reported by the Ridge method were achieved using the filter feature set (i.e., MAE = 0.1527). The Lasso model attained the worst results. The model achieved the same results using the full and filter feature sets (i.e., MAE = 0.1713). tuned RF achieved performance comparable to the FFNN based on the filter features with MAE = 0.1493 based on the complete feature set. tuned SVM utilized the filter feature set to achieve a performance of MAE = 0.1573. Figure 6 shows a comparison between the tuned and not tuned ML models in terms of MAE, MSE, and RMSE. As can be seen, the LSTM model achieved the best results in all metrics.

Table 6 Results of conventional ML with default hyperparameters and averaged time-series data

Models	Features	Evaluation metrics		
		MAE	MSE	RMSE
DT	Full feature set	0.1803	0.0800	0.2556
	Wrapper method	0.1774	0.1220	0.3466
	Filter method	0.1864	0.1256	0.3526
Ridge	Full feature set	0.1717	0.0681	0.2605
	Wrapper method	0.1609	0.0615	0.2477
	Filter method	0.1636	0.0629	0.2481
Lasso	Full feature set	0.1713	0.0641	0.2531
	Wrapper method	0.1713	0.0641	0.2531
	Filter method	0.1734	0.0651	0.2514
RF	Full feature set	0.1580	0.0637	0.2524
	Wrapper method	0.1555	0.0631	0.2510
	Filter method	0.1574	0.0619	0.2469
SVM	Full feature set	0.1553	0.0601	0.2450
	Wrapper method	**0.1492**	**0.0591**	**0.2432**
	Filter method	0.1587	0.0619	0.2433

Table 7 Results of tuned LSTM, conventional ML, and FFNN using averaged time-series

Models	Features	Evaluation metrics		
		MAE	MSE	RMSE
LSTM	Full feature set	0.1531	0.0585	0.2419
	Wrapper method	0.1440	0.0570	0.2387
	Filter method	**0.1375**	**0.0538**	**0.2318**
DT	Full feature set	0.1484	0.0597	0.2443
	Wrapper method	0.1453	0.0624	0.2493
	Filter method	0.1439	0.0557	0.2359
Ridge	Full feature set	0.1576	0.0625	0.2496
	Wrapper method	0.1555	0.0598	0.2444
	Filter method	0.1527	0.0595	0.2439
Lasso	Full feature set	0.1713	0.0641	0.2531
	Wrapper method	0.1733	0.0650	0.2535
	Filter method	0.1713	0.0641	0.2531
RF	Full feature set	0.1538	0.0617	0.2459
	Wrapper method	0.1539	0.0623	0.2494
	Filter method	0.1493	0.0613	0.2475
SVM	Full feature set	0.1573	0.0616	0.2479
	Wrapper method	0.1609	0.0616	0.2420
	Filter method	0.1573	0.0590	0.2429
FFNN	Full feature set	0.1547	0.0586	0.2421
	Wrapper method	0.1492	0.0579	0.2406
	Filter method	0.1441	0.0585	0.2420

- In the third experiment, we studied the role of time series data on the performance of ML models. As discussed, the LSTM model was based on time series data from four time steps: BL, M06, M12, and M18. In this experiment, we flattened the four-time steps, as shown in Fig. 3. The conventional ML models were tuned using grid search techniques and fusion of the flattened time series and BL modality. Table 8 shows the results of the conventional ML models. Note that FFNN was excluded from this experiment because we concentrated on this model in the following experiment. RF achieved the best results based on the filter feature set (i.e., MAE = 0.1476). The best MAE for the DT model was 0.1509 based on the wrapper feature set. The Ridge model achieved the best MAE of 0.1590 based on the filter feature set. The Lasso regressor achieved the same performance using the full and wrapper feature sets (i.e., MAE = 0.1713). Finally, SVM depended on the full feature set to achieve the performance of MAE = 0.1525. By comparing the performance of tuned conventional ML models based on averaged time series with flattened time series, we found that flattening the data decreased the performance of all models. This is because the flattened datasets had a huge number of features compared with the averaged datasets (see Fig. 3). In numbers, assume that we have n features in every time step, and we have t time steps, then after flattening the number of features becomes $n \times t$.

This large number of features is too noisy to be trained properly by conventional ML models. From the previous experiment, we discovered that the flattening of time series data did not help to improve the performance of the tuned conventional ML models. In the fourth experiment, we explored the role of flattened time-series data to enhance the deep FFNN model, see Table 8. In addition, we investigated the performance of deep FFNN based on the baseline visit data only and based on the M18 visit only. Note that the baseline visit had a more significant time gap to the M48 visit compared with the M18 visit. As a result, it was expected that the performance of M18 data would be higher than that of baseline data because as the gap shortens, the model becomes more confident. FFNN achieved the best results based on the flattened time series data and wrapper feature set (i.e., MAE = 0.1452). In contrast to the conventional ML models, the performance of the FFNN model improved based on the flattened time series compared with the averaged time-series experiment. This finding indicates that the deep neural network can learn more complex patterns and deep feature representations from high-dimensional data. Using the baseline time

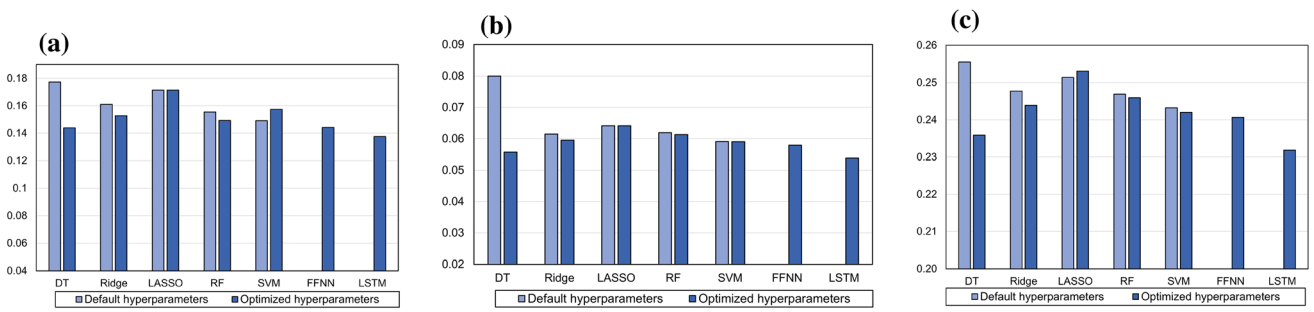

Fig. 6 Comparison of default and tuned hyperparameter settings, **(a)** MAE, **(b)** MSE, and **(c)** RMSE

Table 8 Results of conventional ML and FFNN with grid search and flattened time-series

Models	Features	Evaluation metrics		
		MAE	MSE	RMSE
DT	Full feature set	0.1547	0.0630	0.2500
	Wrapper method	0.1509	0.0610	0.2469
	Filter method	0.1524	0.0608	0.2466
Ridge	Full feature set	0.1709	0.0655	0.2554
	Wrapper method	0.1658	0.0683	0.2608
	Filter method	0.1590	0.0624	0.2498
Lasso	Full feature set	0.1713	0.0641	0.2531
	Wrapper method	0.1713	0.0641	0.2531
	Filter method	0.1714	0.0641	0.2532
RF	Full feature set	0.1550	0.0608	0.2445
	Wrapper method	0.1506	0.0601	0.2450
	Filter method	0.1476	0.0608	0.2463
SVM	Full feature set	0.1525	0.0586	0.2418
	Wrapper method	0.1571	0.0628	0.2502
	Filter method	0.1622	0.0624	0.2495
FFNN + baseline time step	Full feature set	0.1691	0.0638	0.2527
	Wrapper method	0.1689	0.0641	0.2532
	Filter method	0.1710	0.0640	0.2530
FFNN + M18 time step	Full feature set	0.1563	0.0597	0.2443
	Wrapper method	0.1567	0.0580	0.2408
	Filter method	0.1620	0.0621	0.2491
FFNN + flattened time series	Full feature set	0.1547	0.0586	0.2422
	Wrapper method	**0.1452**	**0.0568**	**0.2382**
	Filter method	0.1670	0.0628	0.2506

step to optimize and train the deep FFNN, the model achieved its lowest MAE of 0.1689 based on the wrapper feature set and the lowest MAE 0.0638 based on the complete feature set. Using the data from the M18 visit, we noticed that the performance was improved compared with the baseline visit performance, as expected. The FFNN achieved the best MAE of 0.1563 based on the complete feature set.

Figure 7 provides a summary of all the studied regression models. It compares the best models from all regression experiments, including SVM based on the wrapper feature set and default hyperparameters, RF-based on the filter feature set and tuned feature set, FFNN based on the wrapper feature set and flattened time series, and LSTM based on the filter feature set and tuned architecture. We compared the MAE, MSE, and RMSE of the different models. As it is clearly shown, the deep LSTM model achieved the best scores for all metrics. We can conclude from these results that the LSTM model is able to learn the temporal dependency between time series features and between different features. It is worth noting that these features are lost using the flattened time series with the deep FFNN.

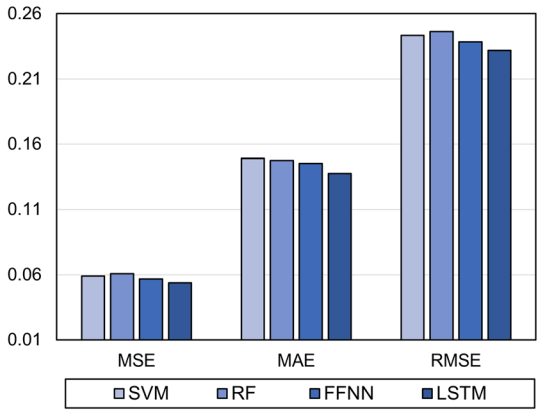

Fig. 7 Proposed two stages framework for hybrid AD diagnosis and conversion prediction

As far as we know, no study in the literature on AD has predicted the exact conversion time of MCI patients. Most studies either model AD progression detection as a binary classification problem [1, 26, 45] (e.g., CN vs. AD [9], MCI vs. AD [10], sMCI vs. pMCI [8, 11]). For example, Westman et al. [68] achieved an accuracy of 91.8% for classifying AD vs. CN based on combined MRI and CSF

data. Their performance dropped to 71.8% for MCI vs. CN. Although these studies were able to achieve outstanding results, their outcomes are not valuable to medical experts because AD is a complex chronic disease whose conversion takes a long time and whose detection is based on multimodal time series data analysis [14, 24, 67]. However, some studies have modelled AD progression as a four-class classification task (CN vs. sMCI vs. pMCI vs. AD) [12, 14, 36, 50, 52, 68–71]. For example, Yao et al. [50] achieved 54.38% accuracy using a hierarchical ensemble and based on baseline data of MRI, age, gender, and MMSE. Nanni et al. [52] achieved 52.92% accuracy using a voting classifier and baseline MRI, age, and MMSE data. Liu et al. [48] achieved an accuracy of 51.8% based on CNN and baseline MRI data. Sorensen et al. [68] achieved 59.10% accuracy using bagging and baseline MRI data, age, gender, and MMSE. Other studies achieved similar accuracies, such as Dimitriadis and Liparas [69] (61.90%), and Jin and Deng [14] (56.25%). This kind of multiclass classification problem is more challenging, and all studies achieved low performance. For example, Jin and Deng [14] achieved an sMCI vs. pMCI accuracy of 60%, but the four-class prediction accuracy was only 30%. Relaxing the four-class problem to a three-class one enhances the performance. For example, Moore et al. [12] achieved 73% accuracy based on an RF classifier and baseline data of MRI and CSs. However, previous studies have ignored the prediction of the exact conversion time of MCI cases. As a result, there are two major challenges in this field of study: (1) to accurately predict the patient's class (CN, MCI, or AD), and (2) to concentrate on the MCI category and predict the exact conversion time of pMCI patients.

5 Limitations and future directions

In this study, we propose a novel framework to predict the patient's AD progression and the exact conversion time for pMCI patients. The study achieved superior results. Based on these results, the proposed framework could be integrated in a real electronic health record (EHR) environment. Figure 8 illustrates the high-level sequence of steps for system development and usage. The figure shows how the deployed system from the proposed model could be integrated in the EHR system in hospitals. The current study has three major limitations that will be handled in our future studies. First, medical experts do not trust machine learning decisions without an accurate and robust explanation [45]. There are many techniques to provide explainability features to a ML model [72] We will extend the current study to provide physicians with explanations in different formats, including feature importance [23], fuzzy rules [73], natural language text [74], and similar cases [75].

Second, we will integrate deep learning and machine learning capabilities to implement hybrid models [67]. This hybridization is expected to improve the diagnosis accuracy of the resulting models. Third, because many AD patients might live in developing countries, where advanced technologies like MRI and PET are not available [25, 76]. Building a decision support system based on cost-effective (bio) markers and lab tests could help detect MCI and AD patients in these poor regions of the world.

6 Conclusion

In this paper, we proposed an DL model for AD detection and pMCI conversion time prediction based on the early fusion of multimodal time-series data. Specifically, the framework was implemented based on an LSTM classification model as a first stage, and then an LSTM regression model as a second stage. The proposed model was able to predict the exact time of AD conversion for MCI patients. To select the best model, we implemented and tuned a collection of ML and DL classifiers and regressors based on a large collection of real prepared cases from the ADNI dataset. Further, we explored different feature selection techniques and different forms of using the time series data. Extensive experiments showed that the LSTM-based classifier achieved the highest CV results using the wrapper-based feature set. In contrast, the LSTM-based regressor achieved the highest results using the filter-based feature set.

It is worth noting that we used a novel method based on the 'divide and conquer' concept. Our two-stage framework was based on a three-class classification task to determine a patient class in the first stage. In the second stage, the model concentrated on MCI patients to decide on their exact progression time. This is the first study to deal with this issue in the AD domain. However, our study still has some limitations. The first limitation is that the model's decisions are not interpretable. A domain expert would not be able to understand why the model took specific decisions. This problem is related to explainability, fairness, and accountability which are at the core of explainable artificial intelligence. In the future, we will extend this work to convert this black-box model into a glass-box one. The second limitation is related to the fusion of other critical modalities, such as PET, symptoms, lab tests, and genetic data. Furthermore, medication modality is cost-effective and could have an important role in AD progression detection. We will explore the role of this modality in the future studies. The fusion of multimodalities will improve the stability of and confidence in the

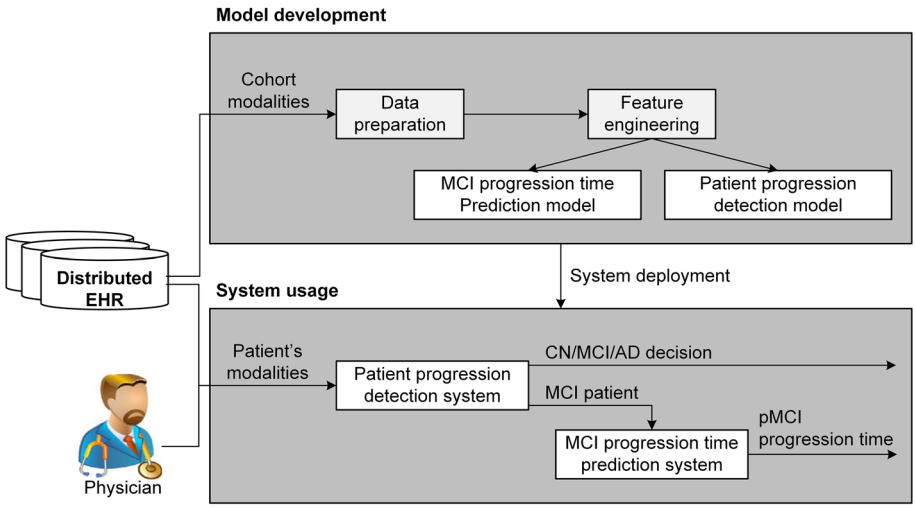

Fig. 8 System development and integration in EHR systems

model. In addition, this will increase medical experts' trust in it. These issues will be handled in our future work. The proposed model can be used to solve other medical and non-medical problems that that are based on time series data and could be formulated as two-stage task. Scientists in other domains can consider the proposed model for their problem solving as a way for applying deep learning algorithms in time series data analysis. The used LSTM model is considered as the best technique for learning deep representations from time series data. In the same time, formulating the problem in two stages relaxes the complexity of a problem and could improves the performance of the model.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00521-022-07263-9.

Author Contributions All authors contributed to the study conception and design. Data collection, preparation, and analysis were performed by Shaker El-Sappagh, Hager Saleh, and Farman Ali. The first draft of the manuscript was written by Shaker El-Sappagh and Hager Saleh. Eslam Amer, Farman Ali, and Tamer ABUHMED contributed to methodology, and writing–review and editing. Tamer ABUHMED contributed to resources, supervision, funding acquisition and project administration. All authors read and approved the manuscript.

Funding This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Creative Consilience Program (IITP-2021-2020-0-01821) supervised by the IITP (Institute for Information \& communications Technology Planning \& Evaluation), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1011198).

Declarations

Conflicts of interest All authors declare that they have no conflicts of interest.

Human and animal rights This research study was conducted retrospectively using human subject data made available by Alzheimer's disease Neuroimaging Initiative (ADNI).

Reproducibility For reproducibility purposes, readers can find the project code at this link: https://github.com/hagersalehahmed/Alzheimer. Because of data privacy, we cannot share the dataset, but a complete description of the used feature set and patient roster IDs are available on Github.

References

1. Ritter K, Schumacher J, Weygandt M, Buchert R, Allefeld C, Haynes J-D (2015) Multimodal prediction of conversion to Alzheimer's disease based on incomplete biomarkers. Alzheimers Dement (Amst) 1(2):206–215
2. Zhang R, Simon G, Yu F (2017) Advancing Alzheimer's research: a review of big data promises. Int J med inform 106:48–56
3. Alzheimer Disease International: World Alzheimer Report 2018 (2018). https://www.alz.co.uk/research/world-report-2018 Accessed 2021
4. Hong X, Lin R, Yang C, Cai C, Clawson K (2020) ADPM: an Alzheimer's disease prediction model for time series neuroimage analysis. IEEE Access 8:62601–62609
5. Iddi S, Li D, Aisen PS, Rafii MS, Thompson WK, Donohue MC (2019) Predicting the course of Alzheimer's progression. Brain Inform 6(1):1–18
6. Lu S, Xia Y, Cai W, Fulham M, Feng DD (2017) Alzheimer's sisease neuroimaging initiative: early identification of mild cognitive impairment using incomplete random forest-robust support vector machine and FDG-PET imaging. Comput Med Imaging Graph 60:35–41
7. Zhou T, Thung K-H, Liu M, Shi F, Zhang C, Shen D (2020) Multi-modal latent space inducing ensemble SVM classifier for early dementia diagnosis with neuroimaging data. Med image anal 60:101630
8. Moscoso A, Silva-Rodríguez J, Aldrey JM, Cortés J, Fernández-Ferreiro A, Gómez-Lado N, Ruibal Á, Aguiar P (2019) Alzheimer's disease neuroimaging initiative: prediction of Alzheimer's disease dementia with MRI beyond the short-term: implications for the design of predictive models. Neuroimage Clin 23:101837

9. Zhang D, Wang Y, Zhou L, Yuan H, Shen D (2011) Alzheimer's disease neuroimaging initiative: multimodal classification of Alzheimer's disease and mild cognitive impairment. Neuroimage 55(3):856–867
10. Hong X, Lin R, Yang C, Zeng N, Cai C, Gou J, Yang J (2019) Predicting Alzheimer's disease using LSTM. IEEE Access 7:80893–80901
11. Filipovych R, Davatzikos C (2010) Alzheimer's disease neuroimaging initiative: semi-supervised pattern classification of medical images: application to mild cognitive impairment (MCI). Neuroimage 55(3):1109–1119
12. Moore PJ, Lyons TJ, Gallacher J (2019) Alzheimer's disease neuroimaging initiative: random forest prediction of Alzheimer's disease using pairwise selection from time series data. PLoS One 14(2):0211558
13. Rallabandi VPS, Tulpule K, Gattu M (2020) Automatic classification of cognitively normal, mild cognitive impairment and Alzheimer's disease using structural MRI analysis. Inform Med Unlocked 18:100305
14. Jin M, Deng W (2018) Predication of different stages of Alzheimer's disease using neighborhood component analysis and ensemble decision tree. J Neurosci Meth 302:35–41
15. Klomp A, Caan MW, Denys D, Nederveen AJ, Reneman L (2012) Feasibility of ASL-based phMRI with a single dose of oral citalopram for repeated assessment of serotonin function. Neuroimage 63(3):1695–1700
16. Yau W-YW, Tudorascu DL, McDade EM, Ikonomovic S, James JA, Minhas D, Mowrey W, Sheu LK, Snitz BE, Weissfeld L et al (2015) Longitudinal assessment of neuroimaging and clinical markers in autosomal dominant Alzheimer's disease: a prospective cohort study. The Lancet Neurol. 14(8):804–813
17. Martí-Juan G, Sanroma-Guell G, Piella G (2020) A survey on machine and statistical learning for longitudinal analysis of neuroimaging data in Alzheimer's disease. Comp Meth Programs in Biomed 189:105348
18. Forouzannezhad P, Abbaspour A, Fang C, Cabrerizo M, Loewenstein D, Duara R, Adjouadi M (2019) A survey on applications and analysis methods of functional magnetic resonance imaging for Alzheimer's disease. J Neurosci Meth 317:121–140
19. Liu L, Zhao S, Chen H, Wang A (2020) A new machine learning method for identifying Alzheimer's disease. Simul Modell Pract Theory 99:102023
20. Cuingnet R, Gerardin E, Tessieras J, Auzias G, Lehéricy S, Habert M-O, Chupin M, Benali H, Colliot O (2010) Alzheimer's disease neuroimaging initiative: automatic classification of patients with Alzheimer's disease from structural MRI: a comparison of ten methods using the ADNI database. Neuroimage 56(2):766–781
21. Wang T, Qiu RG, Yu M (2018) Predictive modeling of the progression of Alzheimer's disease with recurrent neural networks. Scientific Rep 8(1):1–12
22. Wang X, Qi J, Yang Y, Yang P (2019) A survey of disease progression modeling techniques for alzheimer's diseases. In: 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), vol. 1, pp. 1237–1242. IEEE
23. El-Sappagh S, Alonso JM, Islam SR, Sultan AM, Kwak KS (2021) A multilayer multimodal detection and prediction model based on explainable artificial intelligence for Alzheimer's disease. Scientific Rep 11(1):1–26
24. El-Sappagh S, Abuhmed T, Islam SR, Kwak KS (2020) Multimodal multitask deep learning model for Alzheimer's disease progression detection based on time series data. Neurocomputing 412:197–215
25. El-Sappagh S, Saleh H, Sahal R, Abuhmed T, Islam SR, Ali F, Amer E (2021) Alzheimer's disease progression detection model based on an early fusion of cost-effective multimodal data. Fut Gener Comp Syst 115:680–699
26. Zhang D, Shen D (2011) Alzheimer's disease neuroimaging initiative: multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer's disease. Neuroimage 59(2):895–907
27. Tabarestani S, Aghili M, Eslami M, Cabrerizo M, Barreto A, Rishe N, Curiel RE, Loewenstein D, Duara R, Adjouadi M (2020) A distributed multitask multimodal approach for the prediction of Alzheimer's disease in a longitudinal study. NeuroImage 206:116317
28. Alberdi A, Aztiria A, Basarab A (2016) On the early diagnosis of Alzheimer's Disease from multimodal signals: a survey. Artificial intelligence in medicine 71:1–29
29. Moradi E, Pepe A, Gaser C, Huttunen H, Tohka J (2014) Alzheimer's disease neuroimaging initiative: machine learning framework for early MRI-based Alzheimer's conversion prediction in MCI subjects. Neuroimage 104:398–412
30. Pillai PS, Leong T-Y (2015) Alzheimer's disease neuroimaging initiative: fusing heterogeneous data for Alzheimer's disease classification. Stud Health Technol Inform 216:731–735
31. Ewers M, Walsh C, Trojanowski JQ, Shaw LM, Petersen RC, Jack CR Jr, Feldman HH, Bokde AL, Alexander GE, Scheltens P et al (2012) Prediction of conversion from mild cognitive impairment to Alzheimer's disease dementia based upon biomarkers and neuropsychological test performance. Neurobiol Aging 33(7):1203–1214
32. Liu W, Zhang B, Zhang Z, Zhou X-H (2013) Joint modeling of transitional patterns of Alzheimer's disease. PloS One 8(9):75487
33. Huang L, Gao Y, Jin Y, Thung K-H, Shen D (2015) Soft-split sparse regression based random forest for predicting future clinical scores of Alzheimer's disease. In: International Workshop on Machine Learning in Medical Imaging, pp. 246–254. Springer
34. Lee G, Nho K, Kang B, Sohn K-A, Kim D (2019) Predicting Alzheimer's disease progression using multi-modal deep learning approach. Scientific Rep 9(1):1–12
35. Li H, Habes M, Wolk D, Fan Y (2019) A deep learning model for early prediction of Alzheimer's disease dementia based on hippocampal magnetic resonance imaging data. Alzheimer's & Dementia Alzheimer's dise neuroimag Init 15:1059–1070
36. Qiu S, Chang GH, Panagia M, Gopal DM, Au R, Kolachalama VB (2018) Fusion of deep learning models of MRI scans, Mini-Mental State Examination, and logical memory test enhances diagnosis of mild cognitive impairment. Alzheimer's & Dementia: Diagn, Assessment & Dis Monit 10:737–749
37. Forouzannezhad P, Abbaspour A, Li C, Fang C, Williams U, Cabrerizo M, Barreto A, Andrian J, Rishe N, Curiel RE et al (2020) A Gaussian-based model for early detection of mild cognitive impairment using multimodal neuroimaging. J Neurosci Meth 333:108544
38. Cheng B, Liu M, Zhang D, Munsell BC, Shen D (2015) Domain transfer learning for MCI conversion prediction. IEEE Trans Biomed Eng 62(7):1805–1817
39. Wee C-Y, Yap P-T, Shen D (2012) Alzheimer's disease neuroimaging initiative: prediction of Alzheimer's disease and mild cognitive impairment using cortical morphological patterns. Hum Brain Mapp 34(12):3411–3425
40. Weiner MW, Veitch DP, Aisen PS, Beckett LA, Cairns NJ, Cedarbaum J, Donohue MC, Green RC, Harvey D, Jack CR Jr et al (2015) Impact of the Alzheimer's disease neuroimaging initiative, 2004 to 2014. Alzheimer's & Dementia 11(7):865–884
41. Liu F, Zhou L, Shen C, Yin J (2013) Multiple kernel learning in the primal for multimodal Alzheimer's disease classification. IEEE J Biomed Health Inform 18(3):984–990
42. Cho Y, Seong J-K, Jeong Y, Shin SY (2011) Alzheimer's disease neuroimaging initiative: individual subject classification for

Alzheimer's disease based on incremental learning using a spatial frequency representation of cortical thickness data. Neuroimage 59(3):2217–2230
43. Suresh H, Hunt N, Johnson A, Celi LA, Szolovits P, Ghassemi M (2017) Clinical intervention prediction and understanding using deep networks. arXiv preprint arXiv:1705.08498
44. Tian C, Ma J, Zhang C, Zhan P (2018) A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. Energies 11(12):3493
45. Spasov S, Passamonti L, Duggento A, Liò P, Toschi N (2019) Alzheimer's disease neuroimaging initiative: a parameter-efficient deep learning approach to predict conversion from mild cognitive impairment to Alzheimer's disease. Neuroimage 189:276–287
46. Tabarestani S, Aghili M, Shojaie M, Freytes C, Cabrerizo M, Barreto A, Rishe N, Curiel RE, Loewenstein D, Duara R et al. (2019) Longitudinal prediction modeling of alzheimer disease using recurrent neural networks. In: 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), pp. 1–4. IEEE
47. Choi H, Jin KH (2018) Alzheimer's disease neuroimaging initiative: predicting cognitive decline with deep learning of brain metabolism and amyloid imaging. Behav Brain Res 344:103–109
48. Liu M, Zhang J, Adeli E, Shen D (2018) Joint classification and regression via deep multi-task multi-channel learning for Alzheimer's disease diagnosis. IEEE Trans Biomed Eng 66(5):1195–1206
49. Gupta Y, Lama RK, Kwon G-R, Weiner MW, Aisen P, Weiner M, Petersen R, Jack CR Jr, Jagust W, Trojanowki JQ et al (2019) Prediction and classification of Alzheimer's disease based on combined features from apolipoprotein-E genotype, cerebrospinal fluid, MR, and FDG-PET imaging biomarkers. Front Comp Neurosci 13:72
50. Yao D, Calhoun VD, Fu Z, Du Y, Sui J (2018) An ensemble learning system for a 4-way classification of Alzheimer's disease and mild cognitive impairment. J Neurosci Meth 302:75–81
51. Bucholc M, Ding X, Wang H, Glass DH, Wang H, Prasad G, Maguire LP, Bjourson AJ, McClean PL, Todd S et al (2019) A practical computerized decision support system for predicting the severity of Alzheimer's disease of an individual. Expert Syst Appl 130:157–171
52. Nanni L, Lumini A, Zaffonato N (2018) Ensemble based on static classifier selection for automated diagnosis of mild cognitive impairment. J Neurosci Meth 302:42–46
53. Desikan, R.: S egonne F, Fischl B, Quinn BT, Dickerson BC, Blacker D, Buckner RL, Dale AM, Maguire RP, Hyman BT, Albert MS, Killiany RJ, (2006) An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest. Neuroimage 31:968–980
54. MCKHANN G (1984) Report of the NINCDS-ADRDA work group under the auspices of department of health and human service task force on Alzheimer's disease. Neurology 34, 939–944
55. Quinlan J (1993) C4. 5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA
56. Cutler A, Cutler DR, Stevens JR (2012) Random forests. In: Ensemble Machine Learning, pp. 157–175
57. Andrew AM (2001) An introduction to support vector machines and other kernel-based learning methods. Kybernetes
58. Barber D (2012) Bayesian reasoning and machine learning
59. Smola A, Scholkopf B (2004) A tutorial on support vector regression. Stat Comp 14:199–222
60. Wright RE (1995) Logistic regression
61. Hochreiter S (1997) JA1 4 rgen Schmidhuber."Long Short-Term Memory". Neural Computation **9**(8)
62. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artificial Intell Res 16:321–357
63. Ledig C, Schuh A, Guerrero R, Heckemann RA, Rueckert D (2018) Structural brain imaging in Alzheimer's disease and mild cognitive impairment: biomarker analysis and shared morphometry database. Scientific Rep 8(1):1–16
64. Klöppel S, Abdulkadir A, Jack CR Jr, Koutsouleris N, Mourão-Miranda J, Vemuri P (2012) Diagnostic neuroimaging across diseases. Neuroimage 61(2):457–463
65. Klein-Koerkamp Y, Heckemann RA, Ramdeen KT, Moreaud O, Keignart S, Krainik A, Hammers A, Baciu M, Hot P (2014) Alzheimer'sdisease neuroimaging initiative: amygdalar atrophy in early Alzheimer's disease. Curr Alzheimer Res 11(3):239–252
66. Cui R, Liu M (2019) Alzheimer's disease neuroimaging initiative: RNN-based longitudinal analysis for diagnosis of Alzheimer's disease. Comput Med Imag Graphics 73:1–10
67. Abuhmed T, El-Sappagh S, Alonso JM (2021) Robust hybrid deep learning models for Alzheimer's progression detection. Knowl-Based Syst 213:106688
68. Sorensen L, Nielsen M (2018) Alzheimer's Disease Neuroimaging I. Ensemble support vector machine classification of dementia using structural MRI and mini-mental state examination. J Neurosci Methods 302, 66–74
69. Dimitriadis SI, Liparas D, Tsolaki MN (2017) Alzheimer's disease neuroimaging initiative: random forest feature selection, fusion and ensemble strategy: combining multiple morphological MRI measures to discriminate among healhy elderly, MCI, cMCI and alzheimer's disease patients: from the alzheimer's disease neuroimaging initiative (ADNI) database. J Neurosci Meth 302:14–23
70. Maris E, Oostenveld R (2007) Nonparametric statistical testing of EEG-and MEG-data. J Neurosci Meth 164(1):177–190
71. Ebadi A (2017) Dalboni da Rocha JL, Nagaraju DB, Tovar-Moll F., Bramati I., Coutinho G., et al. Ensemble classification of Alzheimer's disease and mild cognitive impairment based on complex graph measures from diffusion tensor images. Front. Neurosci **11**(56), 10–3389
72. xplainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inform Fus **58**, 82–115 (2020)
73. Mencar C, Alonso JM (2019) Paving the Way to Explainable Artificial Intelligence with Fuzzy Modeling. In: Fuller R, Giove S, Masulli F (eds) Fuzzy Logic and Applications. Springer, Cham, pp 215–227
74. Alonso JM, Bugarin A (2019) ExpliClas: Automatic Generation of Explanations in Natural Language for Weka Classifiers. In: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–6
75. Keane MT, Kenny EM (2019) How case-based reasoning explains neural networks: A theoretical analysis of XAI using post-hoc explanation-by-example from a survey of ANN-CBR twin-systems. In: International Conference on Case-Based Reasoning, pp. 155–171. Springer
76. Shoaip N, Rezk A, EL-Sappagh S, Abuhmed T, Barakat S, Elmogy M (2021) Alzheimer's disease diagnosis based on a semantic rule-based modeling and reasoning approach. CMC-Computers Material & Continua 69(3), 3531–3548

Authors and Affiliations

Shaker El-Sappagh[1,2,6] · Hager Saleh[3] · Farman Ali[4] · Eslam Amer[5] · Tamer Abuhmed[6]

✉ Tamer Abuhmed
tamer@skku.edu

[1] Faculty of Computer Science and Engineering, Galala University, Suez 435611, Egypt

[2] Information Systems Department, Faculty of Computers and Artificial Intelligence, Benha University, Banha 13518, Egypt

[3] Faculty of Computers and Artificial Intelligence, South Valley University, Hurghada, Egypt

[4] Department of Software, Sejong University, Seoul, South Korea

[5] Faculty of Computer Science, Misr International University, Cairo, Egypt

[6] College of Computing and Informatics, Sungkyunkwan University, Suwon, South Korea