Multimodal and Longitudinal Approaches to Alzheimer's Disease Classification

Beatriz Oliveira Ferreira

***Abstract*—Alzheimer's disease is a degenerative brain disease, without a known cure. However, its progress can be delayed, when diagnosed in early stages. MRI and PET imaging are precious tools for this purpose, as they can help to identify this disease years before symptoms appear. The present work analyzed how several classifiers performed in different scenarios. These scenarios included binary and multiclass classification (up to 4 classes), predicting the future diagnosis, predicting the current diagnosis having access to data from multiple previous timesteps and predicting whether a subject's diagnosis would get worst. In this last scenario there was an extra obstacle, since the data was highly imbalanced. SMOTE, random undersampling, and a combination of both were the approaches used to deal with the imbalance. The classifiers used were SVM, RF, NN and RNN. In scenarios where more than one timestep was being considered, the input data was adapted to SVM and RF, by concatenating the features of all timesteps. In addition to the standard classifiers, kSVM was proposed and tested. This classifier's objective was to improve the SVM score when there were several timesteps, by breaking up the features by timestep. However, its performance proved to be worse than the one of SVM, which makes this classifier not very promising. A classifier to predict changes of diagnosis was also proposed.**

***Index Terms*—AD, MCI, MRI, RF, RNN, SVM**

I. Introduction

Alzheimer's disease (AD) is a degenerative brain disease, and the most common type of dementia. In the early stages it causes damage to brain cells in parts of the brain involved in cognitive functions, which translates in difficulties with memory, language, problem-solving and other cognitive skills. In later stages, neurons in other parts of the brain are also damaged or destroyed, affecting a person's ability to perform basic body functions like walking or swallowing, which means that people in these stages are bed-bound and need around-the-clock-care. AD is ultimately fatal [1]. Alzheimer's disease prevalence increases with age. Due to the fact that the population in developed countries is aging, the worldwide prevalence of AD is increasing. Therefore, the costs associated with the disease have also increased [1].

Early diagnosis of the disease is very important, since there is still no cure for AD. There are some drugs that can slow or even stop the progression of the disease, but their effectiveness varies from person to person and is limited in duration [1].

Part of identifying dementia in early stages is to identify Mild Cognitive Impairment, or MCI, which is a condition in which an individual has mild changes in thinking abilities, that affects 15-20% of people aged 65 or more. Subjects with MCI can usually carry everyday tasks, but the cognitive changes are noticeable to friends and family [1]. MCI is a major indicator that a subject may develop Alzheimer or other dementia: 32% of individuals with MCI developed AD in the following 5 years [1].

AD's diagnosis is usually a combination of behavioral changes and biomarkers. The behavioral changes are identified by obtaining a medical and family history from the individual, asking a family member to provide input about changes in thinking skills and behavior, and conducting cognitive tests and physical/neurological examinations are some of the tools to provide AD diagnosis. The biomarkers are signs of physical changes in the brain, like the accumulation of beta-amyloid plaques outside neurons and of protein tau inside neurons. The first change contributes to the decay of number of cells by interfering with synapses, and the second one causes the block of transportation of nutrients and other essential molecules to neurons. Other brain changes include inflammation and atrophy [2]. AD (and MCI, with less severity) is also associated with reduced cortical thickness and surface area in a few brain regions associated with cognitive impairment. These changes are identified with the use of PET, MR imaging and CSF biomarkers [2]. These methods can hint that a subject will convert to AD up to 15 years before symptoms appear [1], [2].

The outline of the remaining document is as follows: in section II the classification methods used are explained. Section III discusses imbalanced data and how to deal with it. Section IV has a brief description of the methods used to evaluate the performance of classifiers. In section V, the data in which the experiments were based on is described. In chapter VI, the specific parameters of each classifier are presented. In chapter VII, the results of the different experiments are presented. At last, in chapter VIII, the conclusions of this work are presented, and some suggestions for future work are discussed.

II. Classification methods

A. SVM

A support vector machine (SVM) is a supervised machine learning method for binary classification problems, that can be applied to both linearly separable and non-linearly separable data. It can also be extended to multi-class problems.

Consider a set of L training points, $\boldsymbol{x}_i, i = 1, ..., L$, where each input $\boldsymbol{x}_i$ belongs to $\Re^D$ and to a class $y_i \in \{-1, 1\}$. The main idea behind SVM is to find the optimal hyperplane to separate the two classes in the training data. This hyperplane is the decision boundary. New data is classified simply by checking in which side of the boundary it falls.

If the problem is binary and the data is linearly separable, the hyperplane that divides the two classes is found by solving the optimization problem in equation 1. The dual form of the problem is presented in equation 2, where $H_{ij} \equiv y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j}$. The full demonstration is available in [3].

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{subject to} && y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0, \forall i \end{aligned} \tag{1}$$

$$\begin{gathered} L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{H}\boldsymbol{\alpha} \\ \text{s. t. } \alpha_i \geq 0\ \forall i, \sum_{i=1}^{L} \alpha_i y_i = 0 \end{gathered} \tag{2}$$

Since H_{ij} depends only on the dot product between pairs of points $\boldsymbol{x}_i \boldsymbol{x}_j$, and not on each input $\boldsymbol{x}_i$, the extension of the SVM algorithm to nonlinear classification can be done with relative ease: rather than mapping each training patterns by a map ϕ, it is enough to know $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, where the function k is called the kernel. Therefore there is no need to explicitly compute ϕ [3]. The most common kernels are the polynomial (equation 3), the sigmoid (equation 4) and the radial basis function (RBF) (equation 5).

$$k(x_i, x_j) = (x_i \cdot x_j + a)^b \tag{3}$$

$$k(x_i, x_j) = \tanh(a x_i \cdot x_j - b) \tag{4}$$

$$k(x_i, x_j) = e^{-\frac{1}{2\sigma^2}\|x_i - x_j\|^2} = e^{-\gamma\|x_i - x_j\|^2} \tag{5}$$

SVM can also be used for multi-class classification, however the best way to do this extension is still an on-going research issue. The method used in this work is the one implemented in the scikit-learn library, one-against-one. Another common method is one-against-all [4].

One-against-one trains a model for every pair of classes, so if there are k classes it will train $k(k-1)/2$ classifiers. The decision for a new data point x is made by majority voting: if the SVM for the pair of classes i, j says x is in the i^{th} class, the vote for this class is incremented. Otherwise is the j^{th} class that is incremented.

One-against-all starts by building k different SVM models, where k is the number of different classes. The m^{th} SVM is trained by labeling all patterns of the m^{th} class in the training dataset with $+1$ and all of the other patterns with -1. This leads to having k different decision functions, $f_1(x), f_2(x), ..., f_k(x)$. A new data point x is labeled with the class that has the largest value of the decision function: $x \equiv \text{argmax}_{m=1,...,k}(f_m(x))$.

B. RF

Random forest is an ensemble learning technique, that can be used for both classification and regression. Although it is very simple, it is very powerful, achieving state-of-the-art accuracy in many problems [5].

The base classifier of random forest is the decision tree. Decision trees' structure is made of a root and nodes, which can be internal or leaf nodes. The former correspond to data being internally split and the later correspond to single classes. There are several algorithms to build decision trees; the most used ones are ID3, C4.5 and CART [5]. They all have the same basis: starting with a training set T and a root node, find the best attribute to splitting the node, using a measure of node impurity (can be misclassification error, entropy or Gini index).

Besides decision trees, the random forest algorithm uses the bagging algorithm and the random subspace method.

The random forest algorithm benefits from having multiple tree classifiers, because this compensates for the bias of a single classifier. The strength of random forest classifiers increases with the strength of individual tree classifiers, but decreases with their correlation. So, to improve accuracy, the "randomness" has to minimize the correlation between different trees, while maintaining strength [6]. This is achieved using sampling from a random feature subspace and bagging.

One of the methods used to ensure that the different trees are independent is randomly sampling from the feature subspace [7]. This means that each tree is constructed with only some features, rather than all of the available ones, that are randomly selected. The number of this feature subset is a hyper-parameter of the random forest algorithm.

The other method, bagging or bootstrap aggregation, consists in creating multiple training sets from a single one, by drawing samples with replacement from the original training set [6]. The number of training sets, and therefore of trained trees, is another hyper-parameter of random forests.

At last, the random forest algorithm needs to combine the decisions of the individual trees. It does this by averaging the conditional probability of each class at the leaves [7].

C. NN

Neural networks are made of several units (also called neurons), organized in different layers. The structure of these networks consists of an input layer, one or more hidden layers, and an output layer, with as many units as network outputs.

Each node receives n input signals x_i (with connection weights w_i) which sum to an activation value y (that is, $y = w_i \cdot x_i + b$, where b is the bias of the node). A suitable transfer (or activation) function F transforms it into the output $F(y)$. The knowledge of a network is contained in the connection weights, which assume their values in the training phase. This training is often done using the backpropagation learning algorithm [9].

The most popular activation function in deep learning, for both input and hidden layers, is RELU, which is given by $f(x) = \max(0, x)$. The activation function of the final layer depends on the purpose of the network. For multiclass and single-label classification the softmax function is used. For binary problems a sigmoid function is preferred [8].

The backpropagation algorithm is used to modify the connection weights w_i, in such a way that they minimize the value of the loss function. To do so, the gradient-descent algorithm is used. It starts from a generic data point p_0 and calculates the gradient ∇, which gives the direction in which to move in order to reach the maximum increase (or decrease, if $-\nabla$

is considered). A new point p_1 is found by moving from p_0 a distance of $\eta\nabla$ in the found direction, where η is the learning rate. The gradient is recalculated in the new point p_1, and algorithm continues iteratively until the gradient is smaller than a given threshold. The backpropagation algorithm can be divided into two steps: forward step, where the input data to the network is propagated through all the layers, and then the loss function is computed; and backward step, where the loss is propagated backwards, and the weights are updated appropriately [9]. The loss function that yields better results depends on the problem's characteristics (for example, whether it is regression, binary classification or multiclass classification). Usually, binary or categorical cross-entropy are used.

A recurrent neural network (RNN) is a type of neural network commonly used with sequential data that has an internal loop, that is, it processes sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far. These networks have a feedback architecture, which means that they have memory: they processes sequences by iterating through its elements and maintaining a state containing information relative to what it has seen so far [8]. When training the RNN, the weights of the different layers are adjusted according to the backpropagation algorithm, which can be easily automated, using proper software like *Keras* and *TensorFlow*. However, it is not as simple to find the optimal hyper-parameters, like the architecture of the network (number of layers and how many units per layer), batch size, dropout rate and regularization. To tune hyper-parameters, the data is split into test, train and validation sets, and then cross-validation is used.

In general, when training a neural network the performance of the model starts by improving rapidly: both the test and the validation losses decrease substantially. However, after a certain number of iterations over the training data, the model starts to overfit: the loss on the training data may continue to drop, but it is because the model has lost generalization ability, and is now learning patterns specific to the data at hand. Meanwhile, the loss on the validation data starts increasing [8]. The most straightforward solution to overfitting is to get more training data, although that is not always possible. Other solutions include reducing the network's size, weight regularization and dropout.

Weight regularization is a way of forcing the network weights to be smaller, because larger weights means that the network is more complex. Like it was seen previously, with the network size, a simple explanation leads to better generalization. To apply weight regularization, a cost is added to the loss function of the network. The cost added can be either proportional to the L1 or L2 norm of the weights, which leads to L1 or L2 regularization [8].

Dropout is one of the most effective techniques to deal with overfitting. It is applied to layers, rather than to the whole network. It consists in randomly dropping a fraction of the weights (usually 0.2 to 0.5) of the layer during training. The idea behind this technique is that by adding some noise to the output of the layers the patterns with less significance, which are specific to the training data rather than a general trend, are broken [8].

D. kSVM

A fourth classification method, kSVM, was considered, with the objective of improving the SVM classification for multiple timesteps, as it is a simplified version of what RNN are to NN. The first SVM receives features from the first timestep, and proceeds as described in the SVM subsection, except that it outputs the probability of each class for each subject in the dataset, rather than a single label.

These probabilities are an input to the following SVM, along with the features from the next timestep. The second SVM is similar to the first one. The k SVM, rather than outputting the probability for each class, outputs the final classification for each subject. A schematic for $k = 3$ is presented in figure 1.

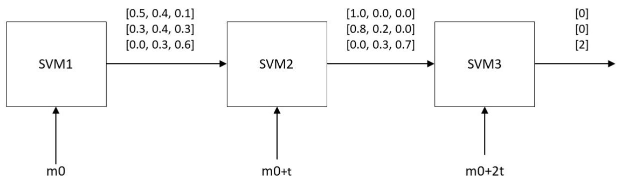


Fig. 1. Structure of the classifier composed by 3SVM. The three sets of numbers represent hypothetical outputs for three input patterns.

E. Training

After building a model, and having the tools to train it, it is necessary to evaluate how well it behaves with new data. In order to do so, rather than training the model with all of the data available, the data is split into three independent sets, called training, test and validation.

The training set is used to train the model, using predetermined hyper-parameters. Then the model is evaluated using the validation set. The hyper-parameters are changed, the model is trained again on the training set and evaluated on the validation set, and so on, until the model has a good performance on the validation set. After the model is trained, it is evaluated on data it has never seen before, the testing set. It is very important to make no changes to the model based on information from the testing set, because it would insert a bias into the model and defeat the purpose of having an independent evaluation of the model [8].

If there are not many data points for training, the validation score may have a high variance depending on the validation slip. Cross-validation is a tool to have a more robust validation.

There are several data resampling strategies, with different degrees of complexity. In practice, k-fold stratified cross-validation is often applied, with k equal to 5 or 10. K-fold cross-validation consists in splitting the data in k partitions, and train k different models. Each model is trained into $k-1$ partitions and evaluated on the remaining one. The validation score of the model is the average of the k scores obtained from each of the k models. This method is represented in figure 2. The stratified part means that the proportions of classes in the different partitions reflect the proportions in the learning set.

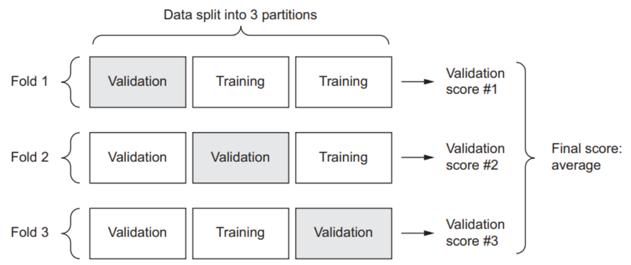


Fig. 2. 3-fold cross validation. Image from Deep Learning with Python [8].

III. Imbalanced data

Although any data set that has unequal distribution between classes can be considered imbalanced, the term imbalanced data is more commonly used when the data set has a significant imbalance, in the order of 100:1, 1,000:1 or even 10,000:1. The main problem when using imbalanced data sets is that they significantly compromise the performance of most standard learning algorithms: they often fail to properly represent the distributive characteristics of the data, and in result provide unfavorable accuracy across the classes of the data. To deal with this problem there are some solutions: resampling, cost-sensitive methods, kernel-based methods, active learning methods, among others [10]. The first two are more common, and achieve state-of-the-art performance in many problems, so they will be explained in more detail. Despite not actually improving the classification, changing the performance metrics can also be a helpful tool when dealing with imbalanced data [10].

Resampling

1) Random oversampling and undersampling: Random oversampling consists in appending to the original data set, S, an extra data set, E, gathered by sampling from the minority class. This means that the number of examples of the minority class in S will increase by $|E|$, which allows to adjust the degree of imbalance. However, if the number of samples in E is too large, the examples of the minority class are repeated many times, which leads to very specific rules, which in turn leads to overfitting [10].

Random undersampling is very similar to random oversampling: rather than appending data from the minority class, it removes data that belongs to the majority class from the original data set, S. The problem with this technique is that removing examples from the majority class may cause the classifier to miss important concepts regarding this class [10].

2) Synthetic samples: The SMOTE (synthetic minority oversampling technique) algorithm [11] creates artificial data points for the minority class based on the similarities between minority examples. To create a synthetic sample, x_{new}, it starts by considering an element of the minority class, x_i and its K nearest neighbors that belong to the same class. Then it randomly chooses one of the K neighbors, $\hat{x_i}$, and a δ between 0 and 1. At last this vector is added to the original sample: $x_{new} = x_i + (\hat{x_i} - x_i) \times \delta$. Figure 3 presents an example of the generation of a synthetic sample using SMOTE, for $K = 6$.

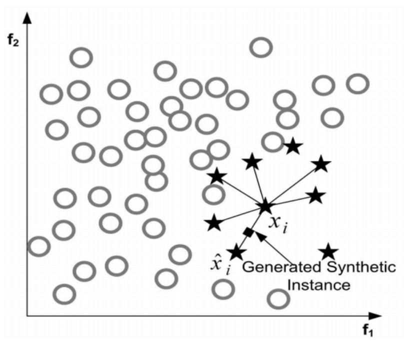


Fig. 3. Example of the creation of a new sample using the SMOTE algorithm. Image from Learning from Imbalanced Data [10].

This method has the same benefits as random oversampling, without the drawbacks. However, the SMOTE algorithm still has some problems, including over generalization and variance [10].

Cost-sensitive methods

Cost-sensitive methods, rather than trying to balance the data set, tackle the imbalanced data problem by using different cost matrices to describe misclassification. These square matrices are composed by as many rows as classes and each cell represents the cost of misclassifying a sample of the class j into the class i, $C(i, j)$ [10].

There are several ways of implementing these methods. The most general ones include dataspace weighting, where the misclassification costs are used to select the best training distribution, and applying cost-minimizing techniques to the combination schemes of ensemble methods. These two methods are commonly used together. However, if there is not an explicit cost matrix available, they cannot be applied.

The alternative is to incorporate cost-sensitive functions directly into classification paradigms, to fit the cost-sensitive framework into these classifiers, but these techniques are often specific to each particular paradigm. For example, in decision trees, cost-sensitive fitting can be applied in the adjustment of the decision threshold, at the split criteria at each node, and in pruning schemes. In neural networks, cost sensitivity can be introduced in four ways: first, cost sensitive modifications can be applied to the probabilistic estimate; second, the neural network outputs can be made cost-sensitive; third, cost-sensitive modifications can be applied to the learning rate; and fourth, the error minimization function can be adapted to account for expected costs [10].

IV. Performance metrics

Performance metrics are key to evaluating a classifier's performance, and to compare different methods. While changing the performance metrics doesn't actually improve the classification, it may provide a better understanding of the classifier. The most frequently used metrics, like accuracy and error rate, provide a simple, and often useful, way of describing a classifier's performance. However, these metrics can be deceiving when dealing with imbalanced data sets [12], which is why in these situations more complex methods, like f1-score, are preferred.

A. Accuracy

Accuracy is the most widely used metric for evaluating performances. It is defined simply as the percentage of correctly classified samples [12]. However, there are shortcomings when dealing with imbalanced data, as it was already mentioned. Accuracy also has problems when dealing with ordinal data: since only correct hits are considered, it does not care about how close/far the misses were.

B. F1-score

F1-score, or f-measure, is a performance metric designed for binary classification, however it can be extended for multiclass classification. It can be interpreted as a weighted average of precision and recall, as presented in equation 6, where β is usually 1. Precision is defined as $TP/(TP+FP)$ and recall is defined as $TP/(TP+FN)$, where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives [12].

$$f1\text{-}score = \frac{(1+\beta)^2 \cdot Recall \cdot Precision}{\beta^2 \cdot Recall + Precision} \tag{6}$$

To apply this method to multiclass classification, the final f1-score equals the weighted average f1-score of each class.

V. Data description

The data used for the experiments comes from the ADNI database [13]. ADNI has data from 3 different phases: ADNI 1, from 2004 to 2009, and ADNI GO/ADNI 2, from 2010 to 2016 (ADNI3 data collection is currently in progress). This work uses data from the 3 available phases (ADNI 1, ADNI 2 and ADNI GO). The number of participants in each one, as well as their clinical status, are presented in table I, where CN means Control Normal, EMCI is Early Mild Cognitive Impairment, LMCI is Late Mild Cognitive Impairment and AD is Alzheimer's Disease. IN ADNI1 there was no distinction between EMCI and LMCI, so there are 400 subjects labeled simply as MCI. ADNI GO only collected data on EMCI subjects.

TABLE I
Number of participants in each ADNI phase, according to their clinical status.

	ADNI1	ADNI GO	ADNI2	Total
CN	200	-	150	350
EMCI	400	200	150	900
LMCI		-	150	
AD	200	-	200	400

In the ADNI database there are different types of data: CSF markers of amyloid-beta and tau deposition, MRI, PET with three tracers (FDG, AV45 and AV1451), cognitive assessments, genetic information and general demographic information [13]. This work focuses on MRI and PET data.

The main advantages of MRI (magnetic resonance imaging) are being noninvasive and that its images have rich and versatile tissue contrast, which can be tuned by different parameters [14].

PET, or positron emission tomography, is an imaging technique that uses radioactive substances, or tracers, to detect functional changes. It is used both in the field of oncology and neuroimaging [15].

MRI and PET data was collected for every subject at the beginning of the study, the baseline (or month 0). Most subjects have measurements for several timesteps: figure 4 presents the number of subjects by visit (every subject that went to visit n went also to visit $n-1$). The time interval between subjects' visits depends on the diagnosis and ADNI phase. A summary of the data collected in function of time and diagnosis is displayed in figure 5.

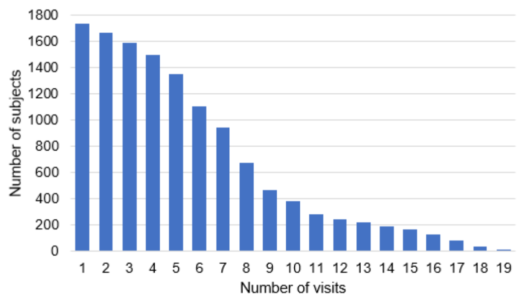
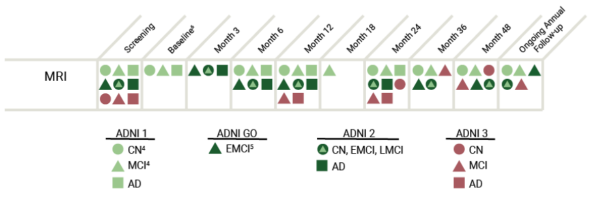
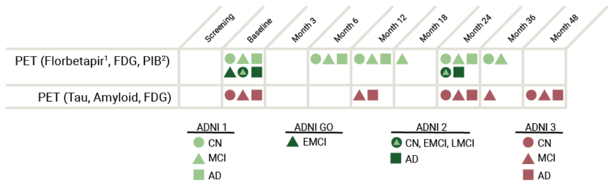


Fig. 4. Number of subjects by visit.



(a) MRI



(b) PET

Fig. 5. Data available in the ADNI database, broken down by diagnosis and ADNI phases, and by timesteps. Figures from [13].

The imaging data used was made available by the TADPOLE challenge [16]. Since it was pre-processed, the data used for classification was already in the form of quantitative numeric measurements.

The pre-processing of MRI scans included correction of gradient non-linearity, B1 non-uniformity scans and peak scans. The relevant regional features, like volume and cortical thickness, were extracted using Freesurfer cross-sectional and longitudinal pipelines. For PET scans, each image is composed by a series of dynamic frames. Its frames were co-registered, averaged across the dynamic range, standardized with respect to the orientation and voxel size and smoothed do produce a uniform resolution of 8mm full-width/half-max [16].

The start point for the remainder of this work was the resulting database.

The features used in the model, rather than selected through feature engineering, were features known to be relevant, based on previous research. More specifically, the features used were the surface area and volume of several regions of interest: ventricles, hippocampus, whole brain, fusiform, middle temporal gyrus and entorhinal cortex. The volumes were normalized with respect to the intracranial volume.

After handling missing data, the MRI dataset had 83 features and the PET dataset had 54 features. The number of subjects in each dataset depended on the timestep being considered. Since PET data from ADNI2 was available only for month 0 and 24, in the following sections the dataset consists only of MRI data, unless stated otherwise.

The maximum number of timesteps considered was 5. There were more timesteps available (figure 5), but they were not evenly distributed.

VI. Experimental design

This section describes the specific parameters and optimization process of each classifier and the methods used in the following experiments.

A. Data preparation

Often, subjects don't have measurements for all timesteps and for all features. The database used for SVM and RF results from the following process: the features that are blank for more than a predetermined threshold of subjects are dropped. Then, all the subjects that are missing information for the remainder features are also dropped. The threshold is chosen so that the number of subjects dropped is minimized, with the condition of keeping at least 10 features.

Before every classification, the features are standardized (for every sample the mean is subtracted and it is divided by the standard deviation).

B. Performance score

All experiments were evaluated using a 5-fold cross-validation process. The final result was the average of the f1-score of each fold. For multiclass problems, the f1-score was computed using the weighted average of the f1-score of each label. For binary problems, the f1-score was simply the one of the positive class.

C. SVM and kSVM

For each fold, the train data is divided into train and validation sets. The train set is used to find the optimal hyper-parameters (kernel, C and γ), through a grid-search (the kernel can be either linear or RBF, C is chosen from $[10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10, 100]$, and for a RBF kernel γ is chosen from $[10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10, 100]$). The model is then trained using these parameters and the training set. The f1-score is obtained with the predictions from the test set.

D. RF

For each fold, the train set is split, to find the optimal hyper-parameters. However, since it would be computationally heavy to optimize a large number of parameters, only the number of trees and the max features are regarded in the optimization process. The number of trees is chosen from [100, 200, 500, 750, 1000, 1500, 2000] and the maximum number of features is chosen from [None,$\sqrt{n}, \log_2(n)$], where n is the number of features.

Other parameters, like maximum depth, minimum samples before splitting and minimum samples per leaf, are not optimized. The model is then trained in the training set and evaluated in the test set.

E. NN

The process starts by converting integer labels into categorical ones, using one hot encoding.

For each fold, two callbacks are used: early stopping and model checkpoint. The early stop one monitors the validation loss. When it has not decreased for a predetermined number of epochs the model stops its training. The model checkpoint monitors the validation accuracy, and saves the model weights that performed better. The model was built using keras library. After some trial and error, the network that showed better results was a sequential network, composed of three layers. The input layer had 25 neurons and the second one had 20. Both have RELU as the activation function. The dropout between these layers was set to 0.2. The number of neurons in the output layer is equal to the number of classes being considered. The output layer is activated by softmax.

VII. Experimental results

A. Predictions for a timestep, given data from that timestep

The first classification problem tackled was to classify subjects into only two classes, AD and CN.

For data corresponding to month 0, there are 1737 subjects. Of those, 865 belong to either AD or CN. After dealing with missing data, there were 852 subjects and 83 features left for MRI data, and 431 subjects and 54 features for PET data. A dataset containing all the common subjects in the MRI and PET database was also considered. This dataset had 424 subjects and 137 features. The labels were converted to numeric values (0 for CN and 1 for AD). F1-scores in percentage and their respective standard deviation are presented in table II.

A second set of results was obtained for month 0, using 3 classes (CN, MCI and AD). The number of features remained equal to what was described in the previous paragraph. The number of subjects in the MRI dataset changed to 1714, in the PET dataset to 890 and in the combined dataset to 877. The average f1-scores in percentage are presented in table III.

A third set of results was obtained for month 0, using 4 classes: MCI was split into EMCI and LMCI. This information is available only for month 0. Table IV presents the average f1-scores.

RF score could be improved by optimizing more parameters. NN could also be improved, by having a systematic optimization of parameters, rather than trial and error. However, both

TABLE II
AVERAGE F1-SCORE FOR CLASSIFYING SUBJECTS INTO CN OR AD, WITH DATA RELATIVE TO MONTH 0.

	PET	MRI	MRI & PET
SVM	83.1 ± 2.8	84.4 ± 3.9	$\mathbf{86.4 \pm 5.5}$
RF	81.9 ± 2.4	83.8 ± 1.4	84.8 ± 5.7
NN	82.8 ± 3.4	83.4 ± 2.6	84.9 ± 5.6

TABLE III
AVERAGE F1-SCORE FOR CLASSIFYING SUBJECTS INTO CN, MCI OR AD, WITH DATA RELATIVE TO MONTH 0.

	PET	MRI	MRI & PET
SVM	51.9 ± 5.4	50.9 ± 1.7	53.7 ± 2.3
RF	51.2 ± 3.4	52.2 ± 2.0	53.1 ± 1.8
NN	53.4 ± 3.8	53.7 ± 2.6	$\mathbf{55.7 \pm 4.1}$

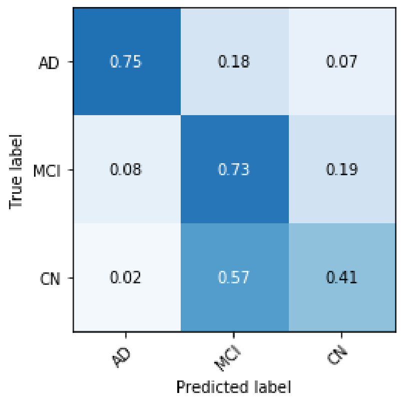


Fig. 6. Confusion matrix obtained for classification of subjects in 3 classes, using MRI and PET imaging. The classifier used was a neural network.

TABLE IV
AVERAGE F1-SCORE FOR CLASSIFYING SUBJECTS INTO CN, EMCI, LMCI OR AD, WITH DATA RELATIVE TO MONTH 0.

	PET	MRI	MRI & PET
SVM	41.9 ± 1.9	41.4 ± 7.1	47.5 ± 1.9
RF	41.1 ± 4.0	43.1 ± 3.4	$\mathbf{48.0 \pm 3.8}$
NN	42.3 ± 5.7	43.0 ± 2.8	47.6 ± 2.1

classifiers are already quite slow. SVM is by far the fastest classifier, and at the same time is the most optimized method. Its results could still be improved, by using a narrower grid to find the optimal parameters.

None of the classifiers under performed or outperformed significantly the others, so all of them were considered in the following sections. Having a combination of MRI and PET measurements always yielded better results than considering a single imaging modality. However, PET data from ADNI2 was available only for month 0 and 24, so in the following sections the dataset consists only of MRI data.

Adding a class between AD and CN lowers the scores significantly. The confusion matrices show that most of the misclassifications are between MCI and CN, as shown in figure 6. This CM was obtained for a combination of MRI and PET features, using a NN as the classifier. It is worth mentioning that approximately half of the subjects in the database belong to the MCI class, so it is expected that this class is favored. When MCI is split into EMCI and LMCI (this data is only available for month 0) there is no longer a bias towards this class. It becomes clear CN and EMCI are often mixed-up, and the same happens for AD and LMCI. This was verified for all the classifiers. As an example, the confusion matrix obtained using SVM, for the database composed of both PET and MRI features, is presented in figure 7.

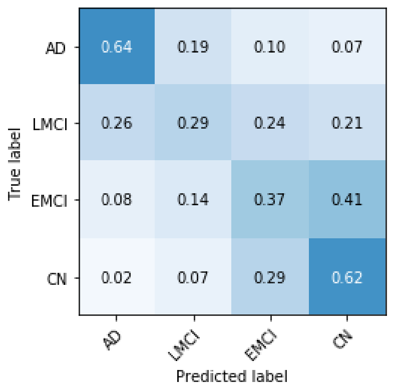


Fig. 7. Confusion matrix obtained for classification of subjects in 4 classes, using MRI and PET imaging. The classifier used was a SVM.

It is likely that classification scores would be improved by filling in missing data rather than just not considering features/subjects with missing data.

B. Predictions for a timestep, given data from the previous timestep

The classification methods used in this section were the same as the ones used in the previous one. The difference was in the training: rather than using data from a given month to obtain a prediction for that same month, the model is trained with features from a previous one and given labels of the given month. Therefore the model has data from the current month, and is trying to predict the future diagnosis.

Below are the results of trying to predict the labels of month 24, using features from month 24 (as a baseline), 12, 6 and 0. The classes considered were CN, MCI and AD. The subjects that converted from one class to another were considered into the class they were converting into (for example, if a subject is classified as "CN to MCI", it is considered MCI). The results are presented in table V. The experiment was repeated using month 12 as reference. The results are presented in table VI.

TABLE V
AVERAGE F1-SCORE FOR PREDICTING THE SUBJECTS' CLASS ON MONTH 24, WITH BASELINE INFORMATION FROM MONTH 24, 12, 6 OR 0.

	Month 0	Month 6	Month 12	Month 24
SVM	52.0 ± 2.6	56.4 ± 4.1	58.2 ± 2.9	57.2 ± 2.6
RF	54.5 ± 2.5	58.3 ± 3.6	$\mathbf{59.8 \pm 2.3}$	58.5 ± 2.6
NN	53.8 ± 1.0	55.3 ± 2.5	56.4 ± 3.2	55.7 ± 3.4

The results show that the predictions get better when using more recent data, which was expected, since the test data used is more similar to the training data. However, unlike what was expected, the results are worse when using features and labels from the same month. For example, in table V the prediction

TABLE VI
AVERAGE F1-SCORE FOR PREDICTING THE SUBJECTS' CLASS ON MONTH 12, WITH INFORMATION FROM MONTH 12, 6 OR 0.

	Month 0	Month 6	Month 12
SVM	51.3 ± 1.9	51.8 ± 3.9	52.2 ± 1.8
RF	50.2 ± 3.4	$\mathbf{55.7 \pm 2.7}$	54.4 ± 0.8
NN	53.7 ± 2.3	54.0 ± 3.2	53.8 ± 5.0

using data from month 0 is worse than the prediction using data from month 6, which in turn is worse than the prediction using data from month 12. However, the prediction using data from month 24 is worse than the one using data from month 12. The results in table VI are similar, which discards the possibility of a statistical outlier. This behavior could be explained by some kind of systematic error, like the time between visitations not being exactly the one considered.

C. Predictions for a timestep, given data from more than one previous timestep

First, subjects were classified into three different classes (AD, MCI and CN) using data from 3 points in time: month 0, 12 and 24.

Two of the classifiers used were the usual SVM and RF. The "memory", or the ability to use information from more than 1 period of time, was created by concatenating features from several timesteps. In this particular case, the information was from month 0, 12 and 24. So, rather than having the usual 83 features, these classifiers had to deal with 249 features (83 features times 3 timesteps). This means that these methods had to look at all the information at once, without having the advantage of "broken-down" sequences.

The third classifier was a RNN. Since memory is its biggest advantage, there was no need to adapt it to receive information from several timesteps. Another upside is that it can easily deal with missing data, by using a masking layer. However, to allow for a more direct comparison, the database used consisted of the same subjects and features as the one used for other classifying methods. The network used consisted of 2 simple RNN layers, each with 32 neurons, dropout equal to 0.1, l1 regularization equal to 0.0001, l2 regularization equal to 0.001 and relu as activation function. The final layer was a dense layer with 3 units, with softmax as activation function. The callbacks Early Stop and Model Checkpoint were used to monitor the validation loss and the validation accuracy, respectively.

The fourth classifier considered was kSVM, with $k = 3$, which means that 3 SVM's were used.

For RF and SVM, the dataset had 249 features and 1016 subjects. The dataset used for both RNN and 3SVM had the same 1016 subjects, with 83 features for each timestep. The subjects were classified into AD, MCI and CN. The results obtained for each method are presented in figure 8.

RF achieved the best score, but the 5-fold validation took 44 minutes to run. SVM took only 89 seconds. RNN and 3SVM took 26 minutes and 6 minutes, respectively.

The same experiment was repeated, but using only the subjects who had measurements for all timesteps (that is,

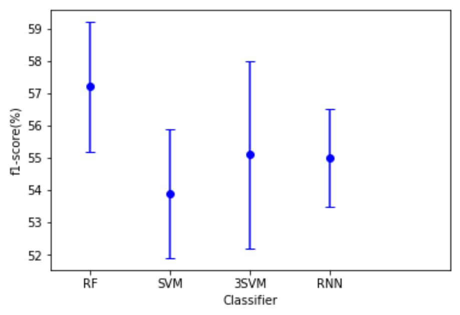


Fig. 8. Average f1-score obtained with different methods for classifying subjects into three classes, using information from months 0, 12, 24.

months 0, 12, 24, 36 and 48). The final database had only 196 subjects and 83 features. Table VII shows the average f1-score and its respective error for each method, obtained through a k-fold cross-validation with $k = 5$. The results of all methods plotted together, so they can be more easily compared, are in figure 9.

Apart from the result of RF for 3 timesteps, all methods have a similar behavior. Figure 9 shows that RNN is the worst method for only two timesteps, but that it improves with each new timestep. Unlike SVM and RF, that can perform well with few training data, RNNs tend to work better with more data. It would be interesting to see if, either with more subjects or with more timesteps, RNN would become the best classifier.

kSVM and SVM have a very similar behavior, despite kSVM being more complex and computationally heavier. Therefore there was no significant advantage in breaking up the features by timesteps before feeding it to kSVM, rather than just using the concatenated version.

TABLE VII
AVERAGE F1-SCORE OBTAINED WITH DIFFERENT METHODS FOR CLASSIFYING SUBJECTS INTO 3 CLASSES, IN FUNCTION OF THE NUMBER OF TIMESTEPS.

# of Timesteps	2	3	4	5
SVM	52.1 ± 4.4	52.0 ± 6.3	52.7 ± 4.6	55.4 ± 7.8
RF	52.3 ± 5.8	56.6 ± 5.2	54.2 ± 5.1	$\mathbf{56.7 \pm 7.1}$
RNN	50.4 ± 8.2	51.2 ± 6.4	52.9 ± 5.5	55.9 ± 4.6
kSVM	52.7 ± 6.7	51.7 ± 2.0	52.2 ± 5.5	54.8 ± 4.1

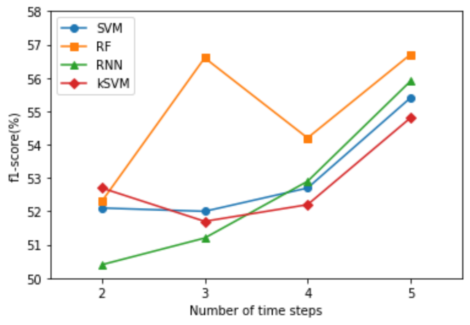


Fig. 9. Comparison of average f1-score obtained with different methods for classifying subjects into three classes, with different number of time steps.

D. Predicting changes of diagnosis

The focus of this experiment was in predicting if a subject's condition got more severe. Using month 0 as starting point and month 24 as the ending point, there were 1020 subjects. In these 2 years, 30 subjects showed improvement (transitioned from MCI to CN or from AD to MCI), so they were removed from the database. 172 subjects transitioned to a worse state, so they were labeled as 1. 818 subjects kept their diagnosis, and were labeled as 0. The features used for SVM, RF and NN were the concatenation of the usual 83 features for months 0 and 24.

This dataset is much more imbalanced than the previous ones, so it was necessary to implement strategies to deal with it: random undersampling (RU), SMOTE and a combination of both. The results are presented in table VIII.

TABLE VIII
AVERAGE F1-SCORE OBTAINED WITH DIFFERENT STRATEGIES TO DEAL WITH CLASS IMBALANCE AND DIFFERENT CLASSIFIERS, FOR CLASSIFYING SUBJECTS AS HAVING CHANGED THEIR DIAGNOSIS BETWEEN MONTH 0 AND MONTH 24.

	Imbalanced	SMOTE	RU	RU & SMOTE
SVM	33.1 ± 5.7	16.4 ± 4.4	40.8 ± 6.3	$\mathbf{40.9 \pm 5.2}$
RF	12.7 ± 9.7	16.6 ± 8.7	27.1 ± 6.8	35.4 ± 2.1
NN	10.8 ± 5.6	30.7 ± 8.2	24.8 ± 3.7	35.3 ± 4.9

Figure 10(a) presents the confusion matrix obtained for SVM without using any approach to deal with the imbalance and figure 10(b) presents the confusion matrix obtained when using SMOTE & RU. Figures 11 and 12 present the same information, for RF and NN, respectively.

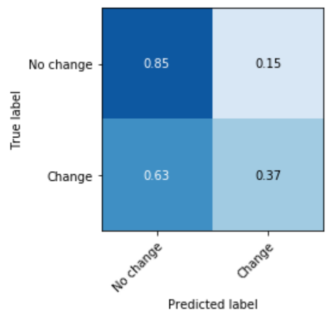
The combination of SMOTE and RU was the best approach for every classifier. As the confusion matrices in figures 10, 11 and 12 show, the accuracy of the minority class as always improved, at the cost of decreasing the accuracy of the majority class. Without dealing with the imbalance, both RF and NN classified almost all test patterns as ”no change”, which means they were initially as good as choosing always the majority class. Using SMOTE & RU, this behavior was altered.

Despite none of the classifiers having achieved a very good performance, SVM and NN reached an accuracy greater than 50% for both classes. To improve the overall performance, tuning the parameters of SMOTE and RU could be attempted. Another strategy could be choosing 2 points in time further apart, which would lead to more subjects converting to a more serious diagnosis, which in turn would result in a more balanced dataset.

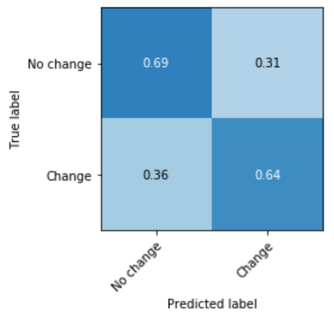
VIII. Conclusions

The present thesis analyzed the performance of different classifiers used to assign subjects into Normal (CN), Mild Cognitive Impairment (MCI) and Alzheimer Disease (AD), by using data from the ADNI database.

In the first experiment of subsection VII-A, the problem of classifying subjects into AD and CN was approached. This was a simple problem, without any real challenge. Its main objective was to test if every classifier was working properly.

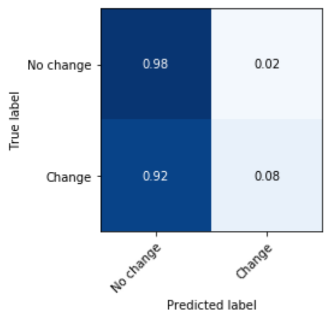


(a) With imbalance

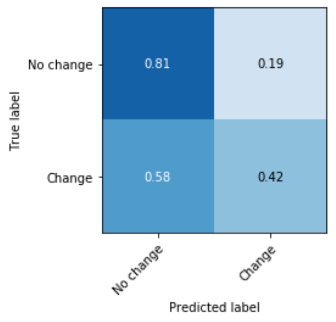


(b) SMOTE & RU

Fig. 10. Confusion matrices obtained for classifying subjects into change/no change, using SVM. The time window considered was from month 0 to month 24.

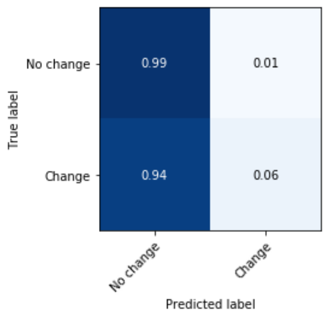


(a) With imbalance

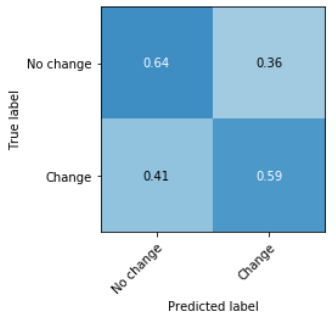


(b) SMOTE & RU

Fig. 11. Confusion matrices obtained for classifying subjects into change/no change, using RF. The time window considered was from month 0 to month 24.



(a) With imbalance



(b) SMOTE & RU

Fig. 12. Confusion matrices obtained for classifying subjects into change/no change, using NN. The time window considered was from month 0 to month 24.

The results were similar for all classifiers and for both imaging modalities, PET and MRI. The best scores were obtained using a combination of PET and MRI features. The maximum f1-score obtained was 86.4.

In the second experiment of the same subsection, 3 classes were considered. As in the previous case, the best results were obtained when using a combination of MRI and PET measurements. The maximum f1-score obtained was 55.7. CN was often classified as MCI. Splitting MCI into EMCI and LMCI, it become clear that MCI is harder to identify. LMCI was often classified as AD and EMCI was often classified as CN. The maximum f1-score obtained using 4 classes was 48.0.

In the following subsection, VII-B, the classification was performed having access to features from a previous timestep and labels from the current one. RF was the classifier that performed better. With the exception of the original timestep, the results are worse for timesteps further away from the original one. However, the decay is not significant for 12 or 18 months. Unlike expected, the results were better using features from the previous timestep than from the current timestep. This effect was observed for both month 12 and 24, which points to it not being a statistical outlier. There could be some kind of systematic error, like the time between visitations not being exactly the one considered.

In section VII-C, information about several points in time was used to classify subjects into 3 classes: CN, MCI and AD. 4 classifiers were used: the well-known SVM, RF and NN and a new one, kSVM. This new classifier used as many SVMs as available timesteps, and the i^{th} SVM received the output from the $i-1^{th}$ SVM and the features belonging to timestep i. In the first experiment, using the bigger dataset, RF obtained the best f1-result: 57.1. However, this method was very slow, and the second experiment showed that this value could be an outlier. RNN and 3SVM obtained a f1-score of 55 and SVM of 54. All of the methods achieved a better f1-score than the ones obtained in section VII-A. In the second experiment of this section, a smaller dataset was used. RF still had the best results, but the value for 3 timesteps seems to be an outlier. All classifiers had a tendency to improve their classification with more timesteps, but RNN's performance was the one that improved the most. The new classifier, kSVM, performed worse than the simple SVM, that received all the data at once, without information regarding which features belonged to each timestep. Since kSVM is computationally heavier, more time-consuming and hard to escalate to many timesteps, it is better to use the classical SVM. Even with a smaller dataset, all of the classifiers achieved average scores significantly higher than the ones obtained in subsection VII-A, for 5 timesteps.

At last, in subsection VII-D, the problem tackled was different: the subjects were now classified as having changed, or not, their diagnosis between 2 points in time. This dataset proved to be very imbalanced. Using the original classifiers, the ”no change” class was chosen most of the time. Using SMOTE and RU, the results improved for all the classifiers, and SVM managed to achieve a f1-score of 40.9. None of the methods proved to be very good to tackle this problem, but SVM and NN reached an accuracy greater than 50% for both classes.

The results obtained in the first experiment using MRI and PET were better than the ones obtained using only MRI. Most results in the present work should improve using a combination of MRI and PET data, rather than just MRI. There could also be some gain in studying the different features, rather than using the ones already used in previous work.

The ordinal nature of the data ended up not being used.

Subjects with missing data and features not present for the majority of subjects were simply not considered, which discarded a significant number of subjects/features. Rather than ”throwing away” information, these ”holes” could be filled, using a number of different imputation approaches.

All of the available parameters were optimized when using SVM. Future work could test if RF results would improve by optimizing more than just 2 parameters, and if NN results would improve by systematically optimizing every parameter, or even the architecture.

In subsection VII-B, the reasons for the classification to be better using past features rather than the features that correspond to the labels are unclear. It could be interesting to see if these differences are due to some systematic error or something else. Using data from later months, there could also be a more thorough analysis on how the error in classification varies in function of time difference between the measurements and the given label.

In the last experiment, corresponding to subsection VII-D, the classifiers had to deal with an imbalanced dataset. The strategies used to deal with it were random undersampling and SMOTE. This strategies were not optimized, as they were implemented using simply the default parameters. The next step could include tuning these parameters and/or experimenting with different approaches. Another possible test is using data from more distant timesteps, so that there are more subjects converting, and the data becomes less imbalanced naturally, rather than adding synthetic samples or removing information.

References

[1] Alzheimer's Association and others. 2018 Alzheimer's disease facts and figures. *Alzheimer's & Dementia,* 14(3):367-429, 2018.

[2] H. Yang, H. Xu, Q. Li, Y. Jin, W. Jiang, J. Wang, Y. Wu, W. Li, C. Yang, X. Li, et al. Study of brain morphology change in Alzheimer's disease and amnestic mild cognitive impairment compared with normal controls. *General psychiatry,* 32(2), 2019.

[3] T. Fletcher. Support Vector Machines Explained. *UCL,* 2008.

[4] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing,* 14(3):199-222, Aug 2004.

[5] M. N. Anyanwu and S. G. Shiva. Comparative analysis of serial decision tree classification algorithms. *International Journal of Computer Science and Security,* 3(3):230–240, 2009.

[6] L. Breiman. Random forests. *Machine Learning,* 45(1):5–32, Oct 2001.

[7] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis & Machine Intelligence,* (8):832–844, 1998

[8] F. Chollet. *Deep learning with Python.* Manning Publications Co., 2018

[9] M. Boden. A guide to recurrent neural networks and backpropagation. *The Dallas project,* 12 2001.

[10] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering,* 21(9):1263–1284, 2009.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research,* 16:321–357, 2002.

[12] L. A. Jeni, J. F. Cohn, and F. De La Torre. Facing imbalanced data–recommendations for the use of performance metrics. In *2013 Humaine association conference on affective computing and intelligent interaction,* pages 245–251. IEEE, 2013.

[13] ADNI - Alzheimer's disease neuroimaging iniciative. URL http://adni.loni.usc.edu/.

[14] K. Möllenhoff, A.-M. Oros-Peusquens, and N. J. Shah. Introduction to the basics of magnetic resonance imaging. In *Molecular Imaging in the Clinical Neurosciences,* pages 75–98. Springer, 2012.

[15] S. Suppiah, M.-A. Didier, and S. Vinjamuri. The who, when, why, and how of PET amyloid imaging in management of Alzheimer's disease – review of literature and interesting images. *Diagnostics,* 9(2):65, 2019.

[16] R. V. Marinescu, N. P. Oxtoby, A. L. Young, E. E. Bron, A. W. Toga, M. W. Weiner, F. Barkhof, N. C. Fox, S. Klein, D. C. Alexander, et al. TADPOLE challenge: Prediction of longitudinal evolution in Alzheimer's disease. *arXiv preprint arXiv:1805.03909,* 2018.