Statistical Modeling and Inference Final Project

Adam Olivares Canal
Berta Canal Simón

December 2023

1 Introduction

Alzheimer's disease is a brain disorder that affects both motor and cognitive function, corresponding to the leading cause of dementia. Since there is no alternative capable of reversing the initial pathological changes associated with the disease, early diagnosis proves advantageous. It helps maintain for longer the patient's functional level, enhancing their quality of life given proper access to treatments and resources (Rasmussen & Langerman, 2019).

Handwriting alterations serve as an early identifiable symptom and a marker of disease onset or progression. Therefore, using handwriting skills to predict a patient's condition is a rational approach in preventing and delaying the disease and subsequent dementia, which becomes increasingly critical considering the global rise in life expectancy.

The input for our algorithms comprises measurements obtained from handwriting samples of both Alzheimer's disease patients and healthy people included in the DARWIN dataset. The aim is to predict whether an individual is affected by Alzheimer and to retrieve the most predictive tasks and features for the identification of said disease. In line with this purpose, the ongoing project uses penalized regressions such as LASSO and Ridge regression, logistic regression using Principal Components and clustering with Gaussian mixture modelling to predict our binary outcome.

The paper aims to contribute to the existing literature by evaluating the success of the aforementioned statistical modelling techniques in capturing the distinctive aspects of handwriting that support the diagnosis of Alzheimer's disease.

2 Related work

There are many previous studies addressed at capturing the distinct aspects of handwriting that support the diagnosis of Parkinson, with datasets such as Parkinson's Disease Handwriting Database (PaHaW) (Drotár et al., 2016) that serves as benchmark data for standardized comparisons across studies. The results extracted from Drotár et al. (2016) through SVM posed the relevance of pressure and kinematic features in assessing subtle characteristics of handwriting and discerning Parkinson patients from healthy controls.

Concerning Alzheimer's disease, the first public dataset corresponds to the ISUNIBA (Pirlo et al., 2015), which contains handwritten features from 29 Alzheimer's disease patients and 12 healthy individuals. The study Pirlo et al. (2015) identifies key kinematic features for early diagnosis using a bagging CART approach.

As a result of the need to expand the studies of Alzheimer's diseases and address the limited sample size of ISUNIBA, Cilia et al. (2022) introduced the DARWIN dataset, specifically designed for early Alzheimer's diagnosis. The paper Cilia et al. (2022) explores a range of machine learning algorithms, among which Random Forest, K-Nearest Neighbor, Linear Discriminant Analysis and Gaussian Naive Bayes.

Subsequent studies emphasizes on an extensive analysis of the classification results from Cilia et al. (2022) with the aim of highlighting common patters along different users and selecting tasks (Gattulli et al., 2023). While these studies adopt the same machine learning algorithms, others propose novel approaches. For instance, Erdogmus and Kabakus (2023) introduces a Convolutional Neural Network (CNN) as a cheap and fast solution, which seems to outperform the accuracy of previous algorithms.

In this study, our goal is to enhance early Alzheimer's detection using penalized regression and dimensionality reduction techniques. This contribution offers a different approach compared

to prevailing papers that predominantly rely on machine learning algorithms.

3 Dataset

The DARWIN dataset comprises 450 features collected from 174 participants: 89 Alzheimer's disease patients and 85 healthy people. It is sourced from the UC Irvine Machine Learning Repository (Fontanella, 2022) and originated from the study *"Diagnosing Alzheimer's disease from online handwriting"* (Cilia et al., 2022). The dataset was generated based on a protocol involving 25 tasks designed with varying complexity, each aimed at specific brain regions. Said tasks are sorted in ascending order in terms of the cognitive demand required and are classified into 4 categories: graphic, copy, memory and dictation tasks. The description of each task are gathered in table A.1

In order to eliminate potential biases, participants were selected to ensure similarity between the two groups concerning age, educational level, occupational type (manual or intellectual) and gender.

These tasks involved handwriting and drawing movements, outlined by 18 distinct features described in Table 1. The dataset includes these features along with a binary variable distinguishing Alzheimer's disease patients from healthy individuals. Additionally, it contains an ID variable excluded from the analysis and there is no presence of missing values.

Features	**Description**
Total Time (TT)	Time spent to perform the entire task.
Air Time (AT)	Time spent performing in-air movements.
Paper Time (PT)	Time spent performing on-paper movements.
Mean Speed on-paper (MSP)	Average speed of on-paper movements.
Mean Speed in-air (MSA)	Average speed of in-air movements.
Mean Acceleration on-paper (MAP)	Average acceleration of on-paper movements.
Mean Acceleration in-air (MAA)	Average acceleration of in-air movements.
Mean Jerk on-paper (MJP)	Average jerk of on-paper movements.
Mean Jerk in-air (MJA)	Average jerk of in-air movements.
Pressure Mean (PM)	Average of the pressure levels exerted by the pen tip.
Pressure Var (PV)	Variance of the pressure levels exerted by the pen tip.
GMRT on-paper (GMRTP)	Generalization of Mean Relative Tremor (MRT) computed for on-paper movements.
GMRT in-air (GMRTA)	Generalization of Mean Relative Tremor (MRT) computed on in-air movements.
Mean GMRT (GMRT)	Average of GMRTP and GMRTA.
Pendowns Number (PWN)	Number of pendowns recorded during the execution of the entire task.
Max X Extension (XE)	Maximum extension recorded along the X axis.
Max Y Extension (YE)	Maximum extension recorded along the Y axis.
Dispersion Index (DI)	Dispersion of the handwritten trace on the entire piece of paper.

Table 1: Features description.

3.1 Preprocessing

The dataset presents substantial correlation among its features, where 111 of them exhibit a correlation exceeding the 90% threshold. The feature correlation is partially addressed leverag-

ing the structured nature of our data which comprises 450 variables that belongs to the results of 18 metrics for each of the 25 distinct handwriting tasks.

The first step of preprocessing is based on a feature selection based on the theoretical meaning of the features and the theoretical design of the tasks. The variable *Total Time (TT)* corresponds to the sum of the time spent to perform both in-air and on-paper movements, therefore it is the linear combination of *Air Time (AT)* and *Paper Time (PT)* with which maintains a high correlation. Furthermore, *Mean GMRT (GMRT)* corresponds to the arithmetic mean between *GMRT on-paper (GMRTP)* and *GMRT in-air (GMRTA)*. Omitting these two covariates leaves us with 400 variables.

Under the assumption that tasks with analogous objectives in targeting specific brain areas yield comparable influences on Alzheimer's disease prediction, it logically follows that the measured features derived from such tasks should manifest a significant correlation. Therefore, the measured features from similar tasks are replaced in the model by the average and difference between them to enhance the ability to both capture the overall trend of their combined effect and reveal nuances in their contribution to the model's predictions. Thus, this transformation aims to reduce multicollinearity without reducing interpretability. After applying these combined variables, the correlations among features decrease, resulting in 67 exhibiting correlations higher than 90%.

The combined tasks correspond as follows: 2 with 3, 4 with 5, 8 with 9, 10 with 12, 11 with 13, and 15 with 16. For example, task 10 consists in copying the word 'fogglio' while task 12 requires to copy the word 'mamma'. The deferring connotation of each word can lead to a distinct performance of the task. Conversely, the analogous design from both tasks suggests a similar predictive capacity in suggesting the existence of the disease.

4 Regression

After having mitigated some of the multicollinearity present in our data in the previous section by omitting and transforming certain features, we are left with 400 variables, which forces us to work in a high-dimensional setting as $p \gg n$. Thus, techniques to model binary outcomes such us the regular logistic regression are not valid options anymore since they become non-identifiable, but if some form of regularization is performed, then it's possible to produce estimates. For instance, penalized likelihood methods like LASSO and Ridge regression are particularly popular, computationally tractable and useful for dimensionality reduction tasks, so we decided to work with them. Moreover, despite having grouped pairs of tests that evaluate almost identical tasks, there are still some highly correlated features in our data due to the specific design of the dataset, as discussed before, so we will also deal with the remaining highly correlated columns of X through LASSO and Ridge regressions.

4.1 LASSO regression

The first method used in our analysis corresponds to the aforementioned Least Absolute Shrinkage and Selection Operator, also known as LASSO, which performs variable selection thanks to the addition of a convex penalty term in the loss function. The Lasso estimator $\boldsymbol{\theta}$ is:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} -\log p(\mathbf{y} \mid \boldsymbol{\theta}) + \lambda |\boldsymbol{\beta}|_1 \tag{1}$$

Where $\lambda \geq 0$ is the regularization parameter and $|\boldsymbol{\beta}|_1 = \sum_{j=1}^{d} \left|\boldsymbol{\beta}_j\right|$ is the L_1 norm.

As stated, the aim of the current project is to predict the Alzheimer's disease via handwriting analysis. Logistic regression is used to predict the binary outcome: patient with Alzheimer's disease ($= 1$) or healthy individual ($= 0$).

In logistic regression $\theta = \boldsymbol{\beta}$, where $\boldsymbol{\beta}$ are the regression coefficients. Therefore, introducing the L_1 norm to the logistic regression results in the following penalized logistic estimator:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} -\mathbf{y}^T X\boldsymbol{\beta} + \sum_{i=1}^{n} \log\left(1 + e^{\mathbf{x}_i^T\boldsymbol{\beta}}\right) + \lambda|\boldsymbol{\beta}|_1 \tag{2}$$

The L_1 norm introduced in the optimization problem for LASSO is a convex function, thus it is ensured that any local optimum is also a global optimum (even when $d > n$). Note that we are dealing with a convex optimization problem because $\log p(\mathbf{y} \mid \boldsymbol{\theta})$ is strictly convex, like any other of the exponential family GLMs.

Given the high amount of features measured in each of the 25 task, it is logical to think that not all of them would be significant for predicting the response, which implies the assumption of β being sparse.

The L_1 constrained region $\{\boldsymbol{\beta} : |\boldsymbol{\beta}|_1 \leq r\}$ for $r > 0$ has sharp edges (cross-polytope), therefore LASSO produces sparse solutions, meaning that its solution $\hat{\boldsymbol{\beta}}$ will have $\hat{\beta}_j = 0$, for many j, which translates into LASSO being able to perform variable selection.

Regarding the penalization parameter λ, two main strategies are considered:

- Set λ via cross-validation to minimize cross-validated prediction error, referred to as LASSO-CV.
- Set λ via BIC to minimize the BIC, referred to as LASSO-BIC. Let $(\hat{\boldsymbol{\beta}}_{\boldsymbol{\lambda}}, \hat{\boldsymbol{\rho}}_{\boldsymbol{\lambda}})$ be the penalized estimator for a given λ:

$$\mathrm{BIC}_\lambda = -2\log p(\mathbf{y} \mid \hat{\boldsymbol{\beta}}_\lambda, \hat{\boldsymbol{\rho}}_\lambda) + |\hat{\boldsymbol{\beta}}_\lambda|_0 \log(n)$$

Therefore, in order to assess the difference in predictive performance of both strategies, two separate models are created, one setting λ via BIC and another one via CV. Both serve different purposes: whereas CV aims at minimizing prediction error (and for prediction overfitting can be benign), BIC is a stricter criteria in terms of variable selection which is model selection consistent and will yield a smaller model.

4.2 Ridge regression

As stated before, the second method used in our analysis corresponds to Ridge regression which differs from Lasso by using the L_2 norm as the penalty term. Therefore, the Ridge estimator $\boldsymbol{\theta}$ is chosen to minimize the penalized sum of squared coefficients:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} -\log p(\mathbf{y} \mid \boldsymbol{\theta}) + \lambda|\boldsymbol{\beta}|_2^2 \tag{3}$$

where $\lambda \geq 0$ is the penalization parameter and $|\boldsymbol{\beta}|_2^2 = \sum_{\boldsymbol{j}=1}^{\boldsymbol{d}} \beta_j^2$ is the squared L_2 norm.

The convex penalty imposed in Ridge regression shrinks the regression coefficients with less contribution to the outcome, but it is not appropriate for model selection because its solution is not sparse. That is, since L_2 constrained region $\{\boldsymbol{\beta} : |\boldsymbol{\beta}|_2 \leq r\}$ for $r > 0$ does not present sharp edges, it does not set coefficients exactly equal 0.

Introducing the penalization in the context of logistic regression results in the following estimator:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} -\mathbf{y}^T X\boldsymbol{\beta} + \sum_{i=1}^{n} \log\left(1 + e^{\mathbf{x}_i^T \boldsymbol{\beta}}\right) + \lambda|\boldsymbol{\beta}|_2^2 \tag{4}$$

The tuning parameter λ is set at the value that reach the minimum mean cross-validated error, which favours its predictive performance.

4.3 Predictive performance

One important remark to be discussed is that in-sample estimates based on the actual observations used to train the model tends to be overoptimistic and, as a consequence, unreliable. Then, given our special interest in producing accurate predictions, a more reasonable approach would be to split n observations into a test and training set. Nonetheless, the estimate would ultimately depend on the used random split, so cross-validation could be used instead to effectively use complete data for training our models, which becomes especially important given our limited number of observations. We therefore evaluate out-sample accuracy with the same metrics by additionally producing 10-fold CV predictions.

Table 2 summarizes the performance metrics computed through cross-validation and the number of selected variables for each of the three models. For our binary outcome target variable, the performance metrics used corresponds to the AUC, False Negative Rate (FNR) and False Positive Rate (FPR).

Model	d	AUC	FNR	FPR
LASSO-CV	44	0.922	0.169	0.094
LASSO-BIC	15	0.745	0.213	0.435
Ridge-CV	400	0.912	0.393	0.094

Table 2: Number of variables and AUC, FNR and FPR obtained via 10-fold cross validation for the regression models.

The highest predictive accuracy is for the LASSO where the penalization parameter λ is set through cross-validation (LASSO-CV). In terms of AUC, LASSO-CV and Ridge-CV attain a similar predictive accuracy but the former presents a considerably lower FNR than the latter. For our particular problem, LASSO-CV outperformed LASSO-BIC. Given that our goal is to forecast, it seems reasonable that the model whose penalization is set with the purpose of minimizing cross-validated prediction error performs better.

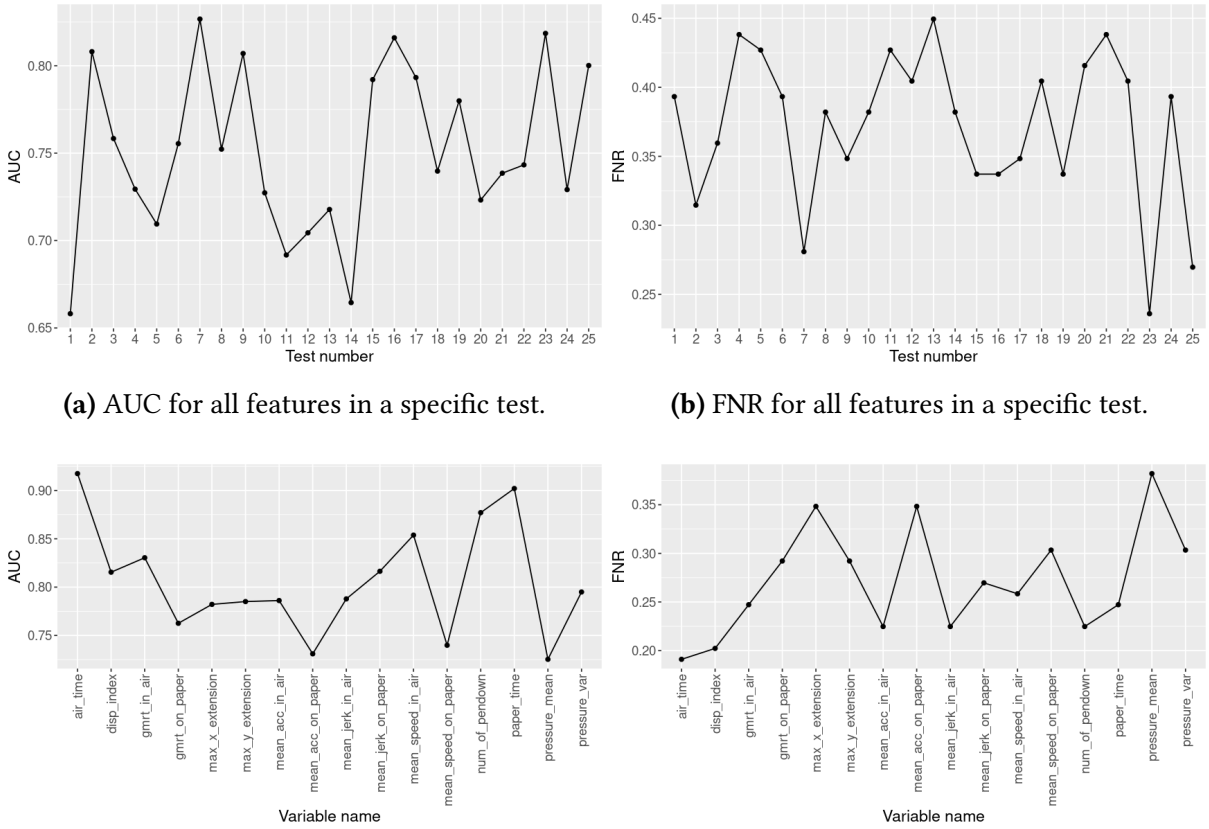
Given the application of the current paper, the False Negative Rate (FNR) metric is a particularly relevant metric, since it would imply classifying as healthy an individual with Alzheimer. Therefore, that patients will not receive treatment, which will cause serious consequences for them since the disease would not be prevented or delayed. Furthermore, it could also be considered the False Positive Rate (FPR), which results in an undesirable situation where a proportion of healthy individuals are categorized as ill. This would subject a healthy patient to unnecessary treatment and its potential side effects. However, since the expected consequences are not that severe, FNR is prioritized in the analysis.

The LASSO-CV consistently shows the best values in both AUC and FNR. Therefore, it corresponds to the model that distinguishes better between Alzheimer patients and healthy individuals, and has the lower proportion of Alzheimer patients classified as healthy. Since our objective is to predict, the selected model would be LASSO-CV.

4.4 Regression for individual tasks and features

Next, the aim is to assess the individual predictive capacity attained per all variables of each task and per a given feature across all tests. On the one hand, for each tasks, it is fitted a LASSO regression using only the features of a specific task (resulting in 25 regressions containing 16 features each). On the other hand, for each feature, a LASSO regression is fitted including a specific feature across all tasks (resulting in 16 regressions of 25 features each). Note that metrics to quantify the quality of predictions were collected using 10 fold cross-validated LASSO.

Note that the two features *Total Time (TT)* and *Mean GMRT (GMRT)* that were previously excluded in the preprocessing are not considered for this analysis.



(a) AUC for all features in a specific test.

(b) FNR for all features in a specific test.

(c) AUC for a specific feature across all tests.

(d) FNR for a specific feature across all tests.

Figure 1: FNR and AUC for all features in a specific test (top panels) and a specific features across all tests (bottom panels).

Regarding the specific LASSO-CV regression for each task, 6 of them outperform the rest of them, obtaining an AUC higher than 0.80, with task 7 achieving the maximum with a value of 0.827. Furthermore, with the exception of three tasks (which lies below the 0.70 threshold), the rest presents an AUC value between 0.70 and 0.80.

Among the tasks which exceeds the 0.80 threshold in terms of AUC, task 7, 23 and 25 also leads in terms of FNR, since they present a value lower than 0.30. With respect to the FNR metric, there are 9 tasks with a FNR that exceeds 0.40, which indicates a bad predictive performance given the relevance of this metric in our context.

It should be noted that the overall predictive performance (measured trough the two metrics of interest) of LASSO-CV regressions considering a specific feature outperform the models that includes all features of a specific task.

In contrast to the tasks case, there is no feature with an AUC lower than 0.70. In fact, the

minimum AUC is reached by the feature *Pressure Mean (PM)* attaining a value of 0.725. In this case, both *Air Time (AT)* and *Paper Time (PT)* presents an AUC exceeding 0.90. The former reaches a value of 0.917, which surpasses the 0.912 attained by Ridge-CV and also presents a similar performance than LASSO-CV (whose AUC correspond to 0.922). Therefore, it should be noted that the predictive ability using only the feature *Air Time (AT)* from 25 tasks is superior to the obtained using 400 features in Ridge-CV and also similar to the obtained using the 44 selected features from LASSO-CV.

Regarding FNR, there is one feature attaining a value higher than 0.35, more especifically, *Pressure Mean (PM)* with a value of 0.382. In contrast, out of the 7 variables whose FNR is below 0.25, the minimum is reached by *Air Time (AT)* with a value of 0.191, whose predictive performance outperforms the other features since it is also the feature reaching the maximum AUC. This value attained by *Air Time (AT)* is also similar to the obtained with LASSO-CV, which suggests a high predictive performance for this specific feature. Among other variables with a high AUC and whose FNR is below 0.25 there are *Pendowns Number (PWN)* and *Paper Time (PT)*. There are also features with a reasonable low FNR but with a low predictive performance in terms of AUC, among which *Mean Acceleration in-air (MAA)* and *Mean Jerk in-air (MJA)*.

An important remark is that although tasks are ordered in terms of complexity, it does not seem to impact the level of predictive performance attained by each task.

5 Unsupervised / Supervised learning

5.1 Sparse PCA

Sparse Principal Component Analysis is a modification to the classical Principal Component Analysis used for dimensionality reduction by creating a set of orthogonal principal components that are linear combinations of the original variables. It modifies the original problem by exploiting the sparsity present in the data. PCA, like many other techniques, fail when $d > n$. Therefore, to produce consistent estimators of our population eigenvectors, we impose a sparse structure on them, thereby allowing for the computation of consistent estimates in high-dimensional settings. That is, the maximal variance characterization of PCA is modified by introducing an L_1 restriction criterion (elastic can be used alternatively) that forces some of the $\boldsymbol{u}$ entries to be zero in the following manner (using singular value representation), where $\boldsymbol{u_1}$ is the right singular vector corresponding to the largest singular value of $\boldsymbol{X} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})'$, which is the $n \times d$ data matrix:

$$\max_{\boldsymbol{u}_1 \boldsymbol{v}_1 : \|\boldsymbol{u}_1\|=1, \|\boldsymbol{v}_1\|=1} \boldsymbol{v}_1' \boldsymbol{X} \boldsymbol{u}_1 \quad \text{subject to} \|\boldsymbol{u}_1\|_1 \leq t. \tag{5}$$

After having applied penalized logistic regression models to obtain coefficients in the previous sections, another reasonable alternative would be to make use of the aforementioned sparse PCA to reduce the dimensionality of our data, which, as explained before, exhibits a good performance under high dimensionality. Then, we will proceed to use a subset of the generated sparse principal components that can capture a decent amount of explained in the dataset. The objective of this reduction of variables is to predict if a patient have Alzheimer (= 1) or not (= 0) with a classical logistic regression, a method that fails to generalize its results when $d > n$. The predictive performance of this model will again be evaluated by computing its AUC, FNR and FPR through 10-fold cross validation.

First, we need to standardize our set of X variables and consider tuning the sparsity assumed in our sparse PCA implementation. In this sense, the library *sparsepca* offers a good implementation in R that will iterate to maximize equation 5. The amount of sparsity assumed can

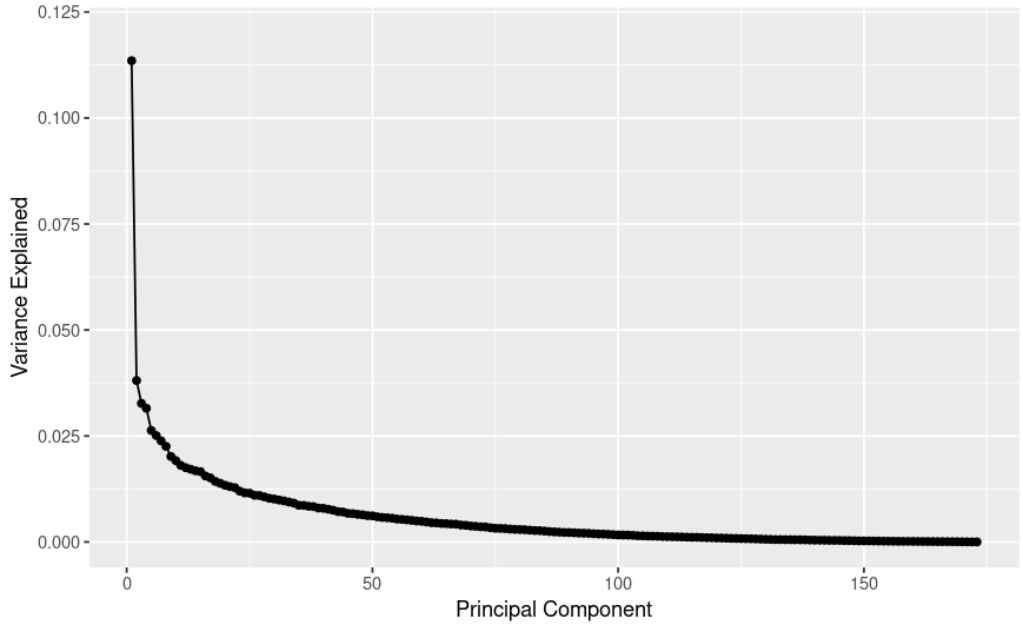


Figure 2: Scree plot depicting the proportion of variance explained by each PC.

be controlled by modifying the parameters alpha and beta, which corresponds to the elastic net sparsity regularizers (a combination of the L_1 and L_2 norms). For our data, a number that performed well in terms of predictive accuracy was the default combination provided by the package (alpha = 1e-04 and beta = 1e-04), but a cross-validation could be used instead to find the best combination of sparsity regularizers in a more orthodox fashion.

Now, we need to address the question of how many principal components we should retain. The first components capture the most variation, so only a few will be retained. A commonly used criterion consists in a scree plot visualization.[1]

Figure 2 doesn't suggest a clear number of PCs to be retained, so we assessed some predictive accuracy metrics for several logistic regressions fitted on a maximum number of first principal components ranging from 1 to 25. We chose the number of PC that deliverd the best result, leading to 5 of them, which only accounted for a 23.6% of the total variance explained. These results can be seen in Table 3.

Model	d	AUC	FNR	FPR
LASSO-CV	44	0.922	0.169	0.094
LASSO-BIC	15	0.745	0.213	0.435
Ridge-CV	400	0.912	0.393	0.094
Logistic (with PCA)	5	0.903	0.157	0.224

Table 3: Number of variables and AUC, FNR and FPR obtained via 10-fold cross validation of all the models seen thus far.

[1] To choose the number of components, we look at the point with the maximum curvature and retain all the components before that point. This test calls for a relative judgment of the variance accounted for by the subsequent components.

5.2 Clustering

Another unsupervised learning technique is clustering, which is intended to discover structures in a given dataset by grouping unlabelled data points in such way that points that are similar in some sense are included in the same group. Such groups are called clusters. Our motivation for using clustering is to observe if it's able to recover the true class assigned to each patient who participated in the clinical assesment, or if, on the contrary, the clustering model captures some underlying pattern in the DARWIN dataset.

There are different approaches that fall under this family of methods, but the ones we are going to considered are the Gaussian mixture models, which assumes that all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. While other hard clustering approaches like K-means offer simple and fast implementations, probabilistic models like Gaussian mixture models can fit clusters with a greater variety of shapes and sizes, which makes it more convenient if we don't have any information about the cluster shapes and sizes since it can generalize better. Notice that K-means is a special case of Gaussian mixture modeling when the weights are equal and the covariance matrices are also the same (and diagonal).

As for the inner mechanism of Gaussian mixture modelling, it involves the mixture of multiple gaussian densities with a weight coefficient π_k that sums up to 1. The likelihood of mixture models is maximized using the EM algorithm and in this way, Gaussian distribution parameters such as mean and variance for each cluster and the weight coefficients of a cluster can be estimated. After learning these parameters for each observation, the posterior probabilities of it belonging to each of the clusters can be derived.

We then implement a Gaussian mixture model on our data with the function 'Mclust'. Gaussian mixture models also require us to choose the type of mixture models and the number of clusters. Given that this procedure is based on maximizing the log likelihood function, we can use an information criteria to choose a model specification. For this task, we used BIC, although ICL was also tried and pointed at the same result. The best BIC value is obtained for VVI (varying volume, varying shape, equal orientation) with 2 clusters.

	gmm.groups 1	gmm.groups 2
True label 0	11	74
True label 1	68	21

Table 4: Crossing of the VVI Gaussian mixture cluster assignments and the true labels of the DARWIN dataset.

Crossing the VVI Gaussian mixture clusters and the true labels of each individual in table 4, we observe that the resulting assignment doesn't separate patients with Alzheimer's from those that are healthy vert accurately. Cluster 1 contains a bigger proportion of individuals with Alzheimer's while cluster 2 groups mostly healthy individuals. In the same way we do with confusion matrices, FPR and FNR can be calculated, obtaining scores of 0.236 and 0.139 respectively.

Furthermore, the results from this method can also be assessed through the corrected Rand index and the Variation of Information index. The Adjusted Rand index, a metric to evaluate the degree of similarity between two cluster assignments, corresponds to 0.396, which is below the acceptable threshold of 0.65. Regarding Variation of Information index, its value is 0.939, which indicates that the distance between two partitions is substantial. Given the poor metrics, it is also suggested that the clustering method is not particularly useful at recovering the true labels of our dependent variable.

6 Discussion

Contributing to Alzheimer's early diagnosis help in preventing and delaying the disease and subsequent dementia. Therefore, throughout our project, the aim has been to obtain a set of models that exhibits good performance to predict whether an individual is affected by Alzheimer and to retrieve the most predictive tasks and features for the identification of said disease. Note that all the models and techniques used in this project were selected considering the high dimensionality inherent in the DARWIN dataset.

The first part of the project performs penalized regression of our data using all features from all tasks and combining those from similar tasks. The most promising methods of this part correspond to LASSO-CV and Ridge-CV, with particular emphasis on the former, which outperforms the latter when it comes to the metrics of interest, particularly in FNR. The performance of both aforementioned penalized regressions is reasonably good, since the strategy chosen for selecting the regularization parameter consists in minimizing the cross-validation prediction error.

In the second part, sparse PCA is used as a dimensionality reduction technique to fit the sparse principal components into a logistic regression. The results from the said method outperforms Ridge-CV and attains similar predictive performance than LASSO-CV, based on AUC and FNR measurements. However, when looking at FPR, LASSO-CV exceeds the value attained by logistic fitted on principal components. On a final note, our findings suggest that using clustering in the given dataset does not successfully recover the true labels of each patient and neither uncover hidden patterns.

Additionally, LASSO-CV regressions fitted on all features for each model individually, demonstrate strong predictive capabilities specifically on tasks 7, 23, and 25. Analogously, evaluating the predictive accuracy of each particular feature involves fitting LASSO regressions with all variables associated with that feature. Surprisingly, the features *Pendowns Number (PWN)*, *Paper Time (PT)*, and *Air Time (AT)* exhibit remarkable predictive performance. Particularly, the latter surpasses Ridge-CV and showcases metrics comparable to LASSO-CV. This implies that certain features can attain a similar predictive performance to models employing all features from all tasks.

The following limitations of the present paper provide avenues for future research. Greater emphasis could be placed on model selection to discern the variables influencing Alzheimer's disease diagnosis, extending beyond those with high predictive performance. Furthermore, the introduction of Bayesian modelling techniques would have allowed for inference on the data.

Overall, the results suggest a potential redesign of the experiment through simplification of tasks and features. This could significantly reduce resource requirements and facilitate obtaining a larger sample size relative to the number of variables. Our project's contribution also extends to demonstrating the capabilities of methods beyond machine learning algorithms, particularly highlighting the robust performance of penalized regression.

References

Bibliographic sources

Cilia, N. D., Gregorio, G. D., Stefano, C. D., Fontanella, F., Marcelli, A., & Parziale, A. (2022). Diagnosing alzheimer's disease from on-line handwriting: A novel dataset and performance benchmarking. *Engineering Applications of Artificial Intelligence*, *111*. https://doi.org/10.1016/j.engappai.2022.104822

Drotár, P., Mekyska, J., Rektorová, I., Masarová, L., Smékal, Z., & Faundez-Zanuy, M. (2016). Evaluation of handwriting kinematics and pressure for differential diagnosis of parkinson's disease. *Artificial Intelligence in Medicine*, *67*. https://doi.org/10.1016/j.artmed.2016.01.004

Erdogmus, P., & Kabakus, A. T. (2023). The promise of convolutional neural networks for the early diagnosis of the alzheimer's disease. *Engineering Applications of Artificial Intelligence*, *123*. https://doi.org/10.1016/j.engappai.2023.106254

Fontanella, F. (2022). DARWIN [DOI: https://doi.org/10.24432/C55D0K].

Gattulli, V., Impedovo, D., Pirlo, G., & Semeraro, G. (2023). Handwriting task-selection based on the analysis of patterns in classification results on alzheimer dataset (V. Dentamaro, M. Parsa, & L. Erdman, Eds.). *3521*, 18–29. https://ceur-ws.org/Vol-3521/paper2.pdf

Pirlo, G., Diaz, M., Ferrer, M. A., Impedovo, D., Occhionero, F., & Zurlo, U. (2015). Early diagnosis of neurodegenerative diseases by handwritten signature analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9281*. https://doi.org/10.1007/978-3-319-23222-5_36

Rasmussen, J., & Langerman, H. (2019). ¡p¿alzheimer's disease – why we need early diagnosis¡/p¿. *Degenerative Neurological and Neuromuscular Disease*, *Volume 9*. https://doi.org/10.2147/dnnd.s228939

Appendix

Task	Description
1	Signature drawing
2	Join two points with a horizontal line, continuously for four times
3	Join two points with a vertical line, continuously for four times
4	Retrace a circle (6 cm of diameter) continuously for four times
5	Retrace a circle (3 cm of diameter) continuously for four times
6	Copy the letters ‘l’, ‘m’ and ‘p’
7	Copy the letters on the adjacent rows
8	Write cursively a sequence of four lowercase letters ‘l’, in a single smooth movement
9	Write cursively a sequence of four lowercase cursive bigram ‘le’, in a single smooth movement
10	Copy the word “foglio”
11	Copy the word “foglio” above a line
12	Copy the word “mamma”
13	Copy the word “mamma” above a line
14	Memorize the words “telefono”, “cane”, and “negozio” and rewrite them
15	Copy in reverse the word “bottiglia”
16	Copy in reverse the word “casa”
17	Copy six words (regular, non-regular, non-words) in the appropriate boxes
18	Write the name of the object shown in a picture (a chair)
19	Copy the fields of a postal order
20	Write a simple sentence under dictation
21	Retrace a complex form
22	Copy a telephone number
23	Write a telephone number under dictation
24	Draw a clock, with all hours and put hands at 11:05 (Clock Drawing Test)
25	Copy a paragraph

Table A.1: Description of the content of each test provided in the paper by Cilia et al. (2022).

Final Project's code

Import libraries and load DARWIN dataset

```
library(glmnet)
library(sigmoid)
library(caret)
library(MASS)
library(ggplot2)
library(dplyr)
library(tidyverse)
library(rstanarm)
library(mombf)
library(ltm)
library(Hmisc)
library(readr)
library(tidyverse)
library(cluster)
library(factoextra)
library(mclust)
library(fpc)
library(stringr)
set.seed(1234)
```

```
darwin <- read.csv("DARWIN.csv", sep = ",", header = TRUE)
darwin <- darwin[,-1]
```

```
darwin_Y <- as.numeric(darwin$class %in% c("P")) #generate a dichotomous variable
darwin_x <- data.frame(darwin[, !colnames(darwin) %in% c("ID", "class")])
```

Data exploration

Count the number of instances of each class (dependent variable). No clas imbalance:

```
ggplot(darwin, aes(x = class)) +
  geom_bar()
```

low variance features.

```
var_test_results <- function(x_data, cutoff){

    coef <- sapply(x_data, function(x) var(x)) #variance
    status <- ifelse(coef >= cutoff, sprintf("above_%s", cutoff),
                     sprintf("below_%s", cutoff))
    data.frame(
    item = names(coef),
    variance = matrix(round(coef,3)),
    status = as.character(status), row.names = c(names(coef))
  )
```

```
}
var_test <- var_test_results(darwin_x, 0.2)

names_list_var_below_cutoff <- row.names(var_test[var_test$variance <= 0.2, ])
```

```
var_test %>% count(status)
```

correlation between x features.

```
correlations <- cor(darwin_x)
highCorr <- findCorrelation(correlations, cutoff = .90, names = TRUE, exact = FALSE)
length(highCorr)

#new_darwin_x <- darwin_x[,!(names(darwin_x) %in% highCorr)]# eliminate by correlation
#names_list_var_below_cutoff
# eliminate by variance
#new_darwin_x <- darwin_x[,!(names(darwin_x) %in% names_list_var_below_cutoff)]
```

```
correlations <- cor(functional_darwin)
highCorr <- findCorrelation(correlations, cutoff = .90, names = TRUE, exact = FALSE)
length(highCorr)
```

Data manipulation

To deal with the problem of high correlation among features, we can transform the most correlated pair of tests in a new set of variables (differences and means) between the variables present in both sets:

drop variables from all tests that we know from theory that have linear dependency ("total_time", "mean_gmrt):

```
drop_darwin_x <- darwin_x %>% dplyr::select(-starts_with(c("total_time", "mean_gmrt")))
```

Get the names of the metrics measured in all tests:

```
# Extract substrings excluding numbers using gsub to obtain unique column names
unique_variable_names <- unique(

  gsub("\\d", "", names(drop_darwin_x))
)
unique_variable_names
```

Function to transform pairs of tests by taking variables and computing their differences and means to deal with colinearity:

```
transformed_darwin_generator <- function(input_df, unique_variable_names,
                                         list_of_pairs, diff = TRUE) {
  stopifnot(is.logical(diff), !is.na(diff)) #warning if diff not boolean


  rowmean_diff_df_gen <- function(test_pairs) {
    test_1 <- test_pairs[1] #element 1 of the vector with a pair
    test_2 <- test_pairs[2] #element 2 of the vector with a pair

    n_obs_individual <- nrow(input_df) #observations
    #number of unique metrics evaluated across experiments
    num_variable_names <- length(unique_variable_names)
```

```
  output_df_1 <- data.frame(matrix(ncol = num_variable_names, nrow = n_obs_individual))
  colnames(output_df_1) <- unique_variable_names #create an empty dataframe

  output_df_2 <- data.frame(matrix(ncol = num_variable_names, nrow = n_obs_individual))
  colnames(output_df_2) <- unique_variable_names #create an empty dataframe

# compute averages
  for (variable_name in unique_variable_names){
    nm_vector <- paste0(variable_name, "", c(test_1, test_2))
    output_df_1[variable_name] <- rowMeans(input_df[nm_vector], na.rm = TRUE)
  }

  mean_names_test_1_2 <- paste0("mean_", unique_variable_names,
                                "_" ,test_1, "_", test_2)
  colnames(output_df_1) <- mean_names_test_1_2

  # compute differences
  if (diff == TRUE){
    for (variable_name in unique_variable_names){
      nm_vector1 <- paste0(variable_name, "", c(test_1))
      nm_vector2 <- paste0(variable_name, "", c(test_2))
      output_df_2[variable_name] <- input_df[nm_vector1] - input_df[nm_vector2]
    }


  diff_names_test_1_2 <- paste0("diff_", unique_variable_names,
                                "_" ,test_1, "_", test_2)
  colnames(output_df_2) <- diff_names_test_1_2

  # merge both metrics

  output_df <- cbind(output_df_1, output_df_2)
  return(output_df)

  } else {
    output_df <- output_df_1
    return(output_df)
  }

  # merge both metrics

  output_df <- cbind(output_df_1, output_df_2)

  return(output_df)
}
### now create the final dataset with the function we created inside this function

#iterate over list of pairs to generate new transformed columns
mean_diff_darwin_x <- data.frame() #empty dataframe to pour values into

  for (pair in list_of_pairs) {
    output <- rowmean_diff_df_gen(pair)
    mean_diff_darwin_x <- data.frame(append(output, mean_diff_darwin_x))
```

```
    }

  # merge with original dataset darwin_x and remove processed columns

  complete_vector <-c()

  for (element in unlist(list_of_pairs)){
    string_vector <- paste0(unique_variable_names, element)
    complete_vector <- append(complete_vector, string_vector)
  }
#remove original processed columns to avoid duplicity
input_df <- input_df %>% dplyr::select(-ends_with(complete_vector))

mean_diff_darwin_x <- cbind(mean_diff_darwin_x, input_df)

return(mean_diff_darwin_x) #return output

}
```

Create a list with vectors including the pairs of tests that contain very similar information which should be transformed into means and differences

```
list_of_pairs <- list(c(2,3), c(4,5), c(8,9), c(10,12), c(11,13), c(15,16))

functional_darwin <- transformed_darwin_generator(drop_darwin_x,
                                  unique_variable_names, list_of_pairs, diff = TRUE)
```

Standardize the created dataframe functional_darwin

```
standardized_functional_darwin <- scale(functional_darwin, scale = TRUE, center = TRUE)
```

Regression

Performance metrics

```
FNR <- function(proba.pred, truth){
  class.pred <- as.numeric(proba.pred > 0.5)
  conf <- table(truth, class.pred)
  print(conf)
  FNR <- conf[2, 1] / sum(conf[2, 1], conf[2, 2])
  return(FNR)
}
```

```
FPR <- function(proba.pred, truth){
  class.pred <- as.numeric(proba.pred > 0.5)
  conf <- table(truth, class.pred)
  print(conf)
  FPR <- conf[1, 2] / sum(conf[1, 1], conf[1, 2])
  return(FPR)
}
```

LASSO-CV (Via cross-validation)

Fitting the model

```
kfoldCV.lasso.logistic <- function(y,x,K=10,seed,criterion='cv') {
## Perform K-fold cross-validation for LASSO regression estimate
  #(lambda set either via cross-val or BIC or EBIC)
## Input
## - y: response
## - x: data.frame with predictors, intercept should not be present
## - K: number of folds in K-fold cross-validation
## - seed: random number generator seed (optional)
## - criterion: the criterion to select the penalization parameter,
  #either cross-val or BIC or EBIC
## Output
## - pred: cross-validated predictions for y
## - ssr: residual sum of squares, sum((y-pred)^2)
  require(glmnet)
  if (!missing(seed)) set.seed(seed)
  subset <- rep(1:K,ceiling(nrow(x)/K))[1:nrow(x)]
  subset <- sample(subset,size=nrow(x),replace=FALSE)
  pred <- double(nrow(x))
  pred_0 <- double(nrow(x))
  cat("Starting cross-validation")
  if (ncol(x)>0) {  #if there are some covariates
    for (k in 1:K) {
        sel <- subset==k
        pred_0[sel] <- mean(y[!sel])
        if (criterion=='cv') {
            fit <- cv.glmnet(x=x[!sel,,drop=FALSE], y=y[!sel],
                             alpha = 1, nfolds=10, family = 'binomial')
            pred[sel] <- predict(fit,newx=x[sel,,drop=FALSE],type='response',
                                 s='lambda.min')
        } else if (criterion=='bic'){
            fit <- lasso.bic.logistic(y=y[!sel],x=x[!sel,,drop=FALSE])
            pred[sel] <- predict(fit$model,newx=x[sel,,drop=FALSE],
                                 type='response', s = fit$lambda.opt)
        } else if (criterion=='ebic'){
            fit <- lasso.bic.logistic(y=y[!sel],x=x[!sel,,drop=FALSE],extended = TRUE)
            pred[sel] <- predict(fit$model,newx=x[sel,,drop=FALSE],
                                 type='response', s = fit$lambda.opt)
        } else { stop("method.lambda not implemented") }
        cat(".")
    }
  } else { #if there are no covariates, just use the intercept
    for (k in 1:K) {
      sel <- subset==k
      pred[sel] <- mean(y[!sel],na.rm=TRUE)
    }
  }
  cat("\n")
  return(list(pred=pred, pred_0= pred_0,ssr=sum((pred-y)^2,na.rm=TRUE)))
}
```

```
set.seed(1234)
t0 <- Sys.time()
fit.lassocv <- cv.glmnet(x=as.matrix(functional_darwin),y=darwin_Y,
                         family ='binomial', nfolds = 10, aplha = 1)
t1 <- Sys.time()
cat('\nTime elapsed: ')
```

```
print(round(t1-t0,3))
```

```
fit.lassocv
```

```
plot(fit.lassocv)
```

```
b.lassocv = as.vector(coef(fit.lassocv, s='lambda.min'))
sum(b.lassocv[-1] != 0)
```

```
rownames(coef(fit.lassocv, s = 'lambda.min'))[coef(fit.lassocv,
                                                   s = 'lambda.min')[,1]!= 0]
```

```
colnames(as.matrix(functional_darwin[,b.lassocv!=0]))
```

Model performance

We get the in-sample auc for the LASSO-CV model:

```
lassocv.pred<- predict(fit.lassocv, newx=as.matrix(functional_darwin),
                       s = fit.lassocv$lambda.min) # NEEDS TO BE FIXED
(auc.lassoCV.training<- pROC::auc(darwin_Y,lassocv.pred))
```

Out-sample prediction

```
cv.pred.lasso.cv <- kfoldCV.lasso.logistic(y=darwin_Y,x=as.matrix(functional_darwin),
                                           K=10,seed=1,criterion="cv")
```

```
(auc.lassocv.test<- pROC::auc(darwin_Y,cv.pred.lasso.cv$pred))
```

```
FNR(cv.pred.lasso.cv$pred,darwin_Y)
```

```
FPR(cv.pred.lasso.cv$pred,darwin_Y)
```

```
mcfadden_pseudo_r2(cv.pred.lasso.cv$pred,cv.pred.lasso.cv$pred_0, darwin_Y)
```

RIDGE

Fitting the model

```
kfoldCV.ridge <- function(y,x,K=10,seed,criterion='cv') {
  ## Perform K-fold cross-validation for Ridge
  # regression estimate (lambda set via cross-val)
  ## Input
  ## - y: response
  ## - x: data.frame with predictors, intercept should not be present
  ## - K: number of folds in K-fold cross-validation
  ## - seed: random number generator seed (optional)
  ## - criterion: the criterion to
  #select the penalization parameter, (only cross-val for now)
  ## Output
```

```
  ## - pred: cross-validated predictions for y
  ## - ssr: residual sum of squares, sum((y-pred)^2)
  require(glmnet)
  if (!missing(seed)) set.seed(seed)
  subset <- rep(1:K,ceiling(nrow(x)/K))[1:nrow(x)]
  subset <- sample(subset,size=nrow(x),replace=FALSE)
  pred <- double(nrow(x))
  cat("Starting cross-validation")
  if (ncol(x)>0) {  #if there are some covariates
    for (k in 1:K) {
      sel <- subset==k
      if (criterion=='cv') {
        fit <- cv.glmnet(x=x[!sel,,drop=FALSE], y=y[!sel],
                         alpha = 0, nfolds=10, family = 'binomial')
        b= as.vector(coef(fit,s='lambda.min'))
        pred[sel] <- b[1] + x[sel,,drop=FALSE] %*% as.matrix(b[-1])
      } else { stop("method.lambda not implemented") }
      cat(".")
    }
  } else { #if there are no covariates, just use the intercept
    for (k in 1:K) {
      sel <- subset==k
      pred[sel] <- mean(y[!sel],na.rm=TRUE)
    }
  }
  cat("\n")
  return(list(pred=pred,ssr=sum((pred-y)^2,na.rm=TRUE)))
}
```

```
fit.ridge= cv.glmnet(x=as.matrix(functional_darwin),
                     y=darwin_Y,family ='binomial', alpha = 0, nfolds=10)
fit.ridge
```

```
b.rigde <- as.vector(coef(fit.ridge, s='lambda.min'))
names(b.rigde) <- c('intercept',colnames(functional_darwin))
sum(b.rigde[-1]!=0)
```

Model performance

Out-sample prediction

```
cv.ridge= kfoldCV.ridge(x=as.matrix(functional_darwin),
                        y=darwin_Y,K=10,seed=1,criterion="cv")
```

```
(auc.ridge.test<- pROC::auc(darwin_Y,cv.ridge$pred))
```

```
FNR(cv.ridge$pred,darwin_Y)
```

```
FPR(cv.ridge$pred,darwin_Y)
```

```
#mcfadden_pseudo_r2(cv.ridge$pred,cv.ridge$pred, darwin_Y)
```

LASSO-BIC

Fitting the model

Now the λ is set via BIC using the function `lasso.bic` from `routines_seminar1.R`.

```
lasso.bic.logistic <- function(y,x,extended=FALSE) {
  #Select model in LASSO path with best BIC (using LASSO regression estimates)
  #Input
  # - y: vector with response variable
  # - x: design matrix
  #
  #Output: list with the following elements
  # - coef: LASSO-estimated regression coefficient with lambda set via BIC
  # - ypred: predicted y
  # - lambda.opt: optimal value of lambda
  # - lambda: data.frame with bic and number of selected variables for each value
  #of lambda
  require(glmnet)
  fit <- glmnet(x=x,y=y,family='binomial',alpha=1)
  pred <- predict(fit,newx=x,type='response')
  n <- length(y)
  p <- colSums(fit$beta!=0) + 1
  if (!extended){
    bic <- -2* colSums(y*log(pred)+(1-y)*log(1-pred)) + log(n)*p
  } else {
    bic <- -2* colSums(y*log(pred)+(1-y)*log(1-pred))
    + log(n)*p + 2*log(choose(ncol(x),p))
  }
  sel <- which.min(bic)
  beta <- c(fit$a0[sel],fit$beta[,sel]); names(beta)[1]= 'Intercept'
  ypred <- pred[,sel]
  ans <- list(model=fit,coef=beta,ypred=ypred,
              lambda.opt=fit$lambda[sel],lambda=data.frame(lambda=fit$lambda,bic=bic,
                                                           nvars=p))
  return(ans)
}
```

To use the BIC criteria, the parameter `extended` is set to `FALSE`.

```
fit.lassobic = lasso.bic.logistic(x = as.matrix(functional_darwin),
                                  y = darwin_Y, extended = FALSE)
b.lassobic = fit.lassobic$coef
names(fit.lassobic)
#round(b.lassobic, 3)
```

The number of non-zero coefficients for LASSO-BIC corresponds to:

```
sum(b.lassobic[-1] != 0)
```

Model performance

We get the in-sample auc for the LASSO-BIC model:

```
(auc.lassoBIC.training<- pROC::auc(darwin_Y,fit.lassobic$ypred))
```

```
cv.pred.lasso.bic <- kfoldCV.lasso.logistic(y=darwin_Y,
                                x=as.matrix(drop_darwin_x),K=10,seed=1,criterion="bic")
```

```r
(auc.lassoBIC.crossvalidated <- pROC::auc(darwin_Y,cv.pred.lasso.bic$pred))
```

```r
FNR(cv.pred.lasso.bic$pred,darwin_Y)
```

```r
FPR(cv.pred.lasso.bic$pred,darwin_Y)
```

```r
mcfadden_pseudo_r2(cv.pred.lasso.bic$pred,
                   cv.pred.lasso.bic$pred_0, darwin_Y)
```

LASSO regressions for individual variables across tests and all variables of a given test

Create subsets of functional_darwin that groups all variables of a given test and the same variable observed across all tests.

```r
#Initialize an empty list to store subsets of the dataframe
dataframe_subsets <- vector("list", 25)  # Assuming you want subsets for numbers 1 to 25

#Group columns by their endings
for (test in 1:25) {
  cols <- grep(paste0("\\D", test, "$"), names(drop_darwin_x))
  dataframe_subsets[[test]] <- drop_darwin_x[, cols, drop = FALSE]
}

dataframe_subsets[[1]]
```

```r
#Initialize an empty list to store subsets of the dataframe
dataframe_subsets_variables <- vector("list")  # Assuming you want subsets
#for numbers 1 to 16

#Group columns by their endings
for (test in unique_variable_names) {
  cols <- grep(paste0(test), names(drop_darwin_x))
  dataframe_subsets_variables[[test]] <- drop_darwin_x[, cols, drop = FALSE]
}

dataframe_subsets_variables[[1]]
```

Use logistic regressions for all the variables of each test individualized

```r
individual_test_performance <- function(x, y){
    indv_test_performance.cv <- data.frame(matrix(ncol = 2, nrow = 25))
    colnames(indv_test_performance.cv) <- c("AUC", "FNR")
  for (i in 1:25){
    #sub_cv.pred.mle<- kfoldCV.mle.logistic(y=y, x=data.frame(x[[i]]),K=10,seed=1)
    sub_cv.pred.lasso <- kfoldCV.lasso.logistic(y=y,x=as.matrix(x[[i]]),K=10,seed=1)
    #lasso
    indv_test_performance.cv[i,1] <- as.numeric(pROC::auc(y,sub_cv.pred.lasso$pred))
    indv_test_performance.cv[i,2] <- as.numeric(FNR(sub_cv.pred.lasso$pred, y))
    #mle
    #indv_test_performance.cv[i,1] <- as.numeric(pROC::auc(y,sub_cv.pred.mle$pred))
    #indv_test_performance.cv[i,2] <- as.numeric(FNR(sub_cv.pred.mle$pred, y))
  }
  return(indv_test_performance.cv)
```

```
}

auc_per_test_mle10cv<- individual_test_performance(dataframe_subsets, darwin_Y)
```

Create a line plot

```
ggplot(data=auc_per_test_mle10cv, aes(x=factor(1:25),
                                      y=auc_per_test_mle10cv[,2], group=1)) +
  geom_line()+
  xlab("Test number") +
  ylab("FNR") +
  geom_point()+
  theme(text = element_text(size=16))
```

Perform logistic regression for each metric across 25 tests to see the predictive ability of same type measurements.

```
individual_test_performance <- function(x, y){
    indv_variable_performance.cv <- data.frame(matrix(ncol = 2, nrow = 16))
    colnames(indv_variable_performance.cv) <- c("AUC", "FNR")
  for (i in 1:16){
    #sub_cv.pred.mle<- kfoldCV.mle.logistic(y=y, x=data.frame(x[[i]]),K=10,seed=1)
    sub_cv.pred.lasso <- kfoldCV.lasso.logistic(y=y,x=as.matrix(x[[i]]),K=10,seed=1)
    #lasso
    indv_variable_performance.cv[i,1] <- as.numeric(pROC::auc(y,sub_cv.pred.lasso$pred))
    indv_variable_performance.cv[i,2] <- as.numeric(FNR(sub_cv.pred.lasso$pred, y))
    #mle
    #indv_variable_performance.cv[i,1] <- as.numeric(pROC::auc(y,sub_cv.pred.mle$pred))
    #indv_variable_performance.cv[i,2] <- as.numeric(FNR(sub_cv.pred.mle$pred, y))
  }
  return(indv_variable_performance.cv)
}

auc_per_variable_mle10cv<- individual_test_performance(dataframe_subsets_variables,
                                                       darwin_Y)
row.names(auc_per_variable_mle10cv) <- unique_variable_names
```

create a line plot

```
ggplot(data=auc_per_variable_mle10cv, aes(x=row.names(auc_per_variable_mle10cv),
                                          y=auc_per_variable_mle10cv[,1], group=1)) +
  geom_line()+
  xlab("Variable name") +
  ylab("AUC") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  geom_point() +
  theme(text = element_text(size=16))
```

```
sub_cv.pred.mle= kfoldCV.lasso.logistic(y=darwin_Y,
                                        x=as.matrix(dataframe_subsets[[13]]),K=10,seed=1)
```

```
(auc.mle.cv <- pROC::auc(darwin_Y,sub_cv.pred.mle$pred))
```

```
FNR(sub_cv.pred.mle$pred,darwin_Y)
```

```
#fit <- bestBIC(darwin_Y ~ ., data = drop_darwin_x, family = "binomial")
#summary(fit)
```

```
#coef(fit) #MLE under top model
#confint(fit) #conf int under top model
```

Sparse PCA and MLE

```
library(sparsepca)
#produce SPCA using  sparsity regularizers
spca.results <- spca(functional_darwin, center = TRUE, scale = TRUE,
                     alpha = 1e-04, beta = 1e-04, tol=1e-05, max_iter = 1000)
```

Plot a screeplot

```
# the eigenvectors can be visualized as eigenfaces, e.g.,
#the first eigenvector (eigenface) is displayed as follows
#summary(spca.results)

### Visualization of variance explained
var_explained <- spca.results$eigenvalues / sum(spca.results$eigenvalues)

summary(spca.results)


qplot(c(1:length(var_explained)), var_explained) +
  geom_line() +
  geom_point(size=0)+
  xlab("Principal Component") +
  xlim(1,25) +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 0.120)

pca_darwin_x <- spca.results$scores[, 1:5]
full_darwin_x_pca <- spca.results$scores[, 1:173]
```

fit MLE

```
logistic_model <- glm(darwin_Y ~., data = data.frame(pca_darwin_x),
                      family = binomial)
# Summarize the model
summary(logistic_model)
# Make predictions
probabilities <- logistic_model %>% predict(data.frame(pca_darwin_x),
                                            type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Model accuracy
mean(predicted.classes == darwin_Y)
```

```
HDRFA::PCA_FN(full_darwin_x_pca, rmax = length(full_darwin_x_pca))
```

mle function for cross-validations

```
kfoldCV.mle.logistic <- function(y,x,K=10,seed) {
## Perform K-fold cross-validation for least-squares regression estimate
## Input
## - y: response
```

```
## - x: data.frame with predictors, intercept should not be present
## - K: number of folds in K-fold cross-validation
## - seed: random number generator seed (optional)
## Output
## - pred: cross-validated predictions for y
## - ssr: residual sum of squares, sum((y-pred)^2)
  if (!missing(seed)) set.seed(seed)
  subset <- rep(1:K,ceiling(nrow(x)/K))[1:nrow(x)]
  subset <- sample(subset,size=nrow(x),replace=FALSE)
  pred <- double(nrow(x))
  pred_0 <- double(nrow(x))
  if (ncol(x)>0) {
    for (k in 1:K) {
      sel <- subset==k
      fit <- glm(y[!sel] ~ ., data=x[!sel,,drop=FALSE], family = 'binomial')
      pred[sel] <- predict(fit, newdata=x[sel,,drop=FALSE],type='response')
      pred_0[sel] <- mean(y[!sel])
    }
  } else {
    for (k in 1:K) {
      sel <- subset==k
      pred[sel] <- mean(y[!sel],na.rm=TRUE)
    }
  }
  return(list(pred=pred, pred_0 =pred_0, ssr=sum((pred-y)^2,na.rm=TRUE)))
}
```

```
cv.pred.mle= kfoldCV.mle.logistic(y=darwin_Y,
                                  x=data.frame(pca_darwin_x),K=10,seed=1)
```

```
(auc.mle.cv <- pROC::auc(darwin_Y,cv.pred.mle$pred))
```

```
FNR(cv.pred.mle$pred,darwin_Y)
```

```
FPR(cv.pred.mle$pred,darwin_Y)
```

Clustering (Gaussian mixtures)

We implement a Gaussian mixture model on our data with the function `Mclust`.

```
mod <- Mclust(standardized_functional_darwin)
summary(mod$BIC)
```

We can use BIC values to choose the type of GMM and optimal number of clusters. The best BIC values is obtained for VVI (varying volume, varying shape, equal orientation) with 2 clusters.

```
plot(mod, what = "BIC", ylim = range(mod$BIC[,-(1:2)], na.rm = TRUE),
     legendArgs = list(x = "bottomleft"))
```

We save our assignments by GMM

```
gmm.groups <- Mclust(standardized_functional_darwin,
                     modelNames = c('VVI'),G=2)$classification
```

Plot of assignations according to posterior probabilities

```r
drmod <- MclustDR(mod, lambda = 1)
plot(drmod, what = "contour")
```

Comparison with true classes

```r
truth.labels <- as.matrix(darwin_Y)
truth <- as.numeric(as.factor(as.matrix(darwin_Y)))
table(truth.labels)
```

```r
compare.gmm.truth <- cluster.stats(
  clustering= mod$classification,
  alt.clustering = truth,
  compareonly = TRUE)

compare.res <- rbind(unlist(compare.gmm.truth))
compare.res <- data.frame(cbind(c('Gaussian Mixtures'),round(compare.res,3)))
colnames(compare.res)[1] <- 'clustering method'
compare.res %>% arrange(desc(corrected.rand))
```

table with true labels and cluster assignments for FPR and FNR

```r
conf_GMM <- table(darwin_Y, gmm.groups)
conf_GMM

FPR_GMM <- conf_GMM[2,2] / sum(conf_GMM[2,2] + conf_GMM[2,1])
FNR_GMM <- conf_GMM[1,1] / sum(conf_GMM[1,1] + conf_GMM[2,1])
FPR_GMM
FNR_GMM
```

Random Forest (DISCARDED IN THE FINAL VERSION)

```r
library(rsample)
library(randomForest)
library(ranger)
library(caret)
library(h2o)

standardized_functional_darwin <- scale(functional_darwin)
```

```r
combined_data <- data.frame(standardized_functional_darwin, darwin_Y)
```

```r
# Loading package
library(caTools)
library(randomForest)
```

```r
set.seed(120)  # Setting seed

# Generating random indices for train and test
train_indices <- sample(nrow(combined_data), 0.8 * nrow(combined_data))  # 70% train
train <- combined_data[train_indices, ]
test <- combined_data[-train_indices, ]

# Separating predictors (x) and target variable (y) for training
```

```r
x_train <- subset(train, select = -darwin_Y)
y_train <- train$darwin_Y
```

```r
library(randomForest)

# Fitting a random forest model
model <- randomForest(x = x_train, y = as.factor(y_train), ntree = 500)

# Summary of the model
print(model)
```

predict probabilities using test train split

```r
# Extracting probabilities for class 1
predicted_probs <- predict(model, test, type = "prob")[, 2]

# Compute AUC
roc_obj <- pROC::roc(test$darwin_Y, predicted_probs)
auc <- pROC::auc(roc_obj)
print(paste("AUC:", auc))

# Compute False Negative Rate (FNR)
threshold <- 0.5  # Threshold for class prediction
predictions <- ifelse(predicted_probs > threshold, 1, 0)
conf_matrix <- table(Actual = test$darwin_Y, Predicted = predictions)
fnr <- conf_matrix[2, 1] / sum(conf_matrix[2, ])
print(paste("FNR:", fnr))
```

```r
# Get feature importance
model <- randomForest(x = x_train, y = as.factor(y_train), ntree = 500)
feature_importance <- randomForest::importance(model)
sorted_importance <- feature_importance[order(-feature_importance[, "MeanDecreaseGini"]),
                                        ]
print(sorted_importance)
```

kfold cross validation as a substitute of train-test split

```r
kfoldCV.randomforest.logistic <- function(y,x,K=10,seed) {
## Perform K-fold cross-validation for least-squares regression estimate
## Input
## - y: response
## - x: data.frame with predictors, intercept should not be present
## - K: number of folds in K-fold cross-validation
## - seed: random number generator seed (optional)
## Output
## - pred: cross-validated predictions for y
## - ssr: residual sum of squares, sum((y-pred)^2)
  if (!missing(seed)) set.seed(seed)
  subset <- rep(1:K,ceiling(nrow(x)/K))[1:nrow(x)]
  subset <- sample(subset,size=nrow(x),replace=FALSE)
  pred <- double(nrow(x))
  pred_0 <- double(nrow(x))
  if (ncol(x)>0) {
    for (k in 1:K) {
      sel <- subset==k
```

```
      #fit <- glm(y[!sel] ~ ., data=x[!sel,,drop=FALSE], family = 'binomial')
      fit <- randomForest(x = x[!sel,,drop=FALSE],
                          y = as.factor(y[!sel]), ntree = 500)
      pred[sel] <- predict(fit, newdata=x[sel,,drop=FALSE],
                           type='response')
      pred_0[sel] <- mean(y[!sel])
    }
  } else {
    for (k in 1:K) {
      sel <- subset==k
      pred[sel] <- mean(y[!sel],na.rm=TRUE)
    }
  }
  return(list(pred=pred, pred_0 =pred_0,
              ssr=sum((pred-y)^2,na.rm=TRUE)))
}
```

Random Forest CV prediction metrics

```
cv.pred.RandomForest <- kfoldCV.randomforest.logistic(y=darwin_Y,
                                                      x=data.frame(functional_darwin),
                                                      K=10,seed=1)
```

Out-sample metrics

```
(auc.randomForest.cv <- pROC::auc(darwin_Y,cv.pred.RandomForest$pred))
```

```
cv.pred.RandomForest$pred -1
```

```
FNR(cv.pred.RandomForest$pred -1,darwin_Y)
```

```
FPR(cv.pred.RandomForest$pred -1,darwin_Y)
```