34th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, Macao, China, 2022

DOI: https://doi.org/10.1109/ICTAI56018.2022.00038

scikit-fda: Computational Tools for Machine Learning with Functional Data

Carlos Ramos-Carreño*‡, José Luis Torrecilla†§, Yujian Hong* and Alberto Suárez*¶
*Department of Computer Science †Department of Mathematics
Universidad Autónoma de Madrid, Madrid, Spain
ORCID: ‡0000-0003-2566-7058, §0000-0003-3719-5190, ¶0000-0003-4534-0909

***Abstract*—Machine learning from functional data poses particular challenges that require specific computational tools that take into account their structure. In this work, we present *scikit-fda*, a Python library for functional data analysis, visualization, preprocessing, and machine learning. The library is designed for smooth integration in the Python scientific ecosystem. In particular, it complements and can be used in combination with *scikit-learn*, the reference Python library for machine learning. The functionality of *scikit-fda* is illustrated in clustering, regression, and classification problems from different areas of application.**

***Index Terms*—functional data analysis, data visualization, machine learning, Python toolbox**

I. Introduction

In many application domains, the data available for learning consist of functions [1]. Examples are meteorology [2], speech analysis [3], spectroscopy [4], and medicine [5], among others. Because of their continuous nature, the analysis of functional data presents specific difficulties. In general, they are infinite-dimensional. Consequently, some quantities, such as probability densities and the likelihood function, which are central in statistical learning, are ill-defined [6]. Furthermore, new types of problems arise when dealing with these types of data. For instance, empirical observations can be contaminated by noise. Smoothing techniques are thus required to uncover the underlying functions, which are often assumed to be continuous and differentiable [1]. Misalignements can also occur when repeated observations are made. If such is the case, registration techniques may be needed to correct for such distortion [7]. In the functional context, variable selection methods are especially important, not only for efficiency, but also for interpretability: they can be used to identify impact points, which capture the most relevant information to explain the phenomenon at hand [8]. Finally, for machine learning, standard algorithms, which are typically formulated in a multivariate setting, need to be adapted, and new ones designed so that they can deal with, and take advantage of the specific characteristics of functional data.

The goal of this paper is to address a representative set of machine learning problems with functional data, including clustering, regression, and classification, from different areas of application. They serve to illustrate the special properties and the difficulties posed by the analysis of these types of data. The study is carried out using *scikit-fda* [9], a Python package that provides a comprehensive set of tools for functional data analysis. The package is designed for smooth integration in the Python scientific ecosystem. In particular, it can be readily used in combination with *scikit-learn* library [10] to address machine learning problems with functional characteristics. The package is fully open source, released under a 3-Clause BSD license and accepting contributions in its GitHub repository https://github.com/GAA-UAM/scikit-fda.

For the sake of reproducibility, the code used in this article is available at https://fda.readthedocs.io/ictai-examples, as part of the documentation of the package, and can be executed in the cloud using the link in the bottom part of each example. Import statements are ommited in the code for clarity.

The structure of the paper is as follows: In Section II, the framework for learning with functional data is introduced. Section III contains an analysis of meteorological data with a number of visualization tools and unsupervised (functional PCA, clustering) methods. In Section IV, a supervised problem illustrates derivatives, basis expansions, regression and variable selection. The last example in Section V illustrates some preprocessing tools exclusive to functional data, such as smoothing and registration, as well as classification methods.

II. Machine learning with functional data

In the functional setting, the data available for induction are functions of a continuous parameter. Assuming, for the sake of concreteness, that this parameter is time, the learning sample consists of a set of trajectories $\{x_n(t),\ t \in \mathcal{T} \subset \mathbb{R}\}_{n=1}^{N}$, where $x_n(t)$ is the nth observation in the sample. In supervised learning, the target labels $\{y_n\}_{n=1}^{N}$ are also known.

Because of their infinite-dimensional nature, functional data cannot be handled directly. Therefore, they need to be represented in a manageable form, without losing their functional essence. To this end, the two main approaches are discretization and basis representation. The first one consists in representing each function by its values at some grid points $\boldsymbol{t} = (t_1, \ldots, t_M) \in \mathcal{T}^M$, i.e., $x(\boldsymbol{t}) = (x(t_1), \ldots, x(t_M))$. Typically, this is the format in which functional processes are recorded in actual measurements. Note that M should be sufficiently large to preserve the functional character of the data. For simplicity, from now on, we assume a regular grid, common for all observations in the sample [11].

Alternatively, each observation can be represented by the coefficients of an expansion in a proper functional basis, such as Fourier, B-splines, or principal components [1]. To make

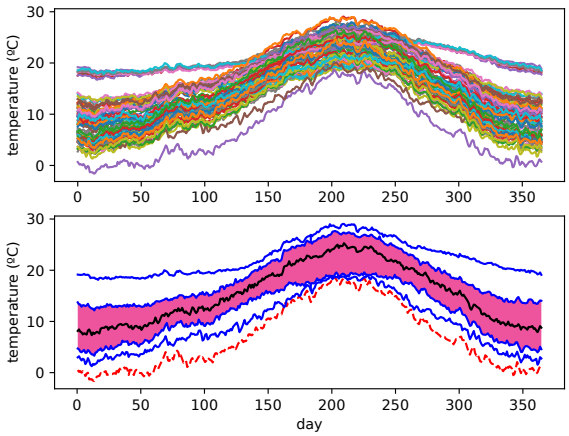


Fig. 1. Functional temperature observations from the AEMET dataset (up) as well as its corresponding boxplot (bottom).

computations feasible, the series, which is in general infinite, is truncated to the B first elements. Thus, the function $x(t)$ is approximated by $x(t) \approx \sum_{b=1}^{B} c_b \phi_b(t)$ where c_b is the expansion coefficient of $\phi_b(t)$, the bth basis function. The package *scikit-fda* include both discretized and basis expansion representations by means of the classes `FDataGrid` and `FDataBasis`, respectively.

III. Meteorological data: data visualization, clustering, and functional PCA

We now use a dataset of annual temperatures from Spain to illustrate some of the functionalities offered by *scikit-fda* for visualization, clustering and functional principal component anaysis (FPCA). The data are taken from the State Meteorological Agency of Spain (AEMET) and contain several meteorological observations (temperature, precipitation and wind speed) at $N = 73$ weather stations throughout Spain. These indicators are measured daily ($M = 365$) between 1980 and 2009 and averaged over years. We can download the data directly with *scikit-fda* by using the following code. The result is an object of class `FDataGrid` which contains the functional data in a discretized form.

```
X, _ = fetch_aemet(return_X_y=True)
```

This is an example of the *scikit-fda* tools for fetching datasets, based on the package *scikit-datasets* [12].

Now, we select the temperature curves (the first coordinate function) and plot them in the upper part of Figure 1.

```
X = X.coordinates[0]
X.plot()
```

We can see that all stations present the typical behaviour of the northern hemisphere, with higher temperatures in summer that descend during winter. However, temperatures at some stations are atypical with respect to the majority: a purple curve with a significantly lower temperature than the others, and a set of flatter curves with warmer winters. In fact, these are outliers of magnitude and shape, respectively. The first one belongs to the Navacerrada station, at 1894 meters in height near a ski resort, while the others correspond to stations in the Canary Islands, known for their subtropical climate.

One way to detect and visualize magnitude outliers is to use the functional boxplot proposed in [13].

```
Boxplot(
    X, depth_method=ModifiedBandDepth(),
).plot()
```

The resulting plot is in the bottom part of Figure 1. This is an extension of the classical univariate boxplot to the functional case. In pink, the central envelope of the data contains the deepest 50% of observations. The outlying envelope, bounded by the most external blue curves, separates the typical trajectories from magnitude outliers such as Navacerrada, in red.

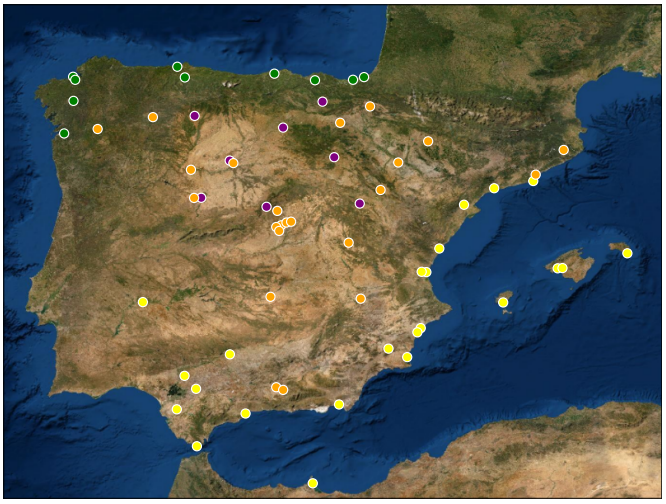
The centrality, or depth, of a curve is quantified by statistical depths measures. For instance, the deepest observation in a dataset corresponds to the median (depicted in black in Figure 1), while outliers have depth values tending to zero. There are many proposals of functional depths, each of which defines different median and envelopes, and verifies different properties [14]. In *scikit-fda* are available integrated depths [15], as well as the band and the modified band depths [16]. The last one is used in the previously shown boxplot.

The functional boxplot is not very suitable for detecting shape outliers. To this end, the magnitude-shape plot [17] and the outliergram [18], both available in the library, can be used. All these visual tools can be combined and used interactively.

We now center our attention in grouping the stations by climate using the annual temperatures. This is a clustering problem that can be addressed adapting the classical k-means algorithm to the functional setting, using a distance between functions. The library *scikit-fda* also provides hierarchical clustering, and fuzzy c-means, a variant of k-means in which each observation has a degree of membership to each cluster. The package offers a variety of functional distances. Here, we use the L^2 distance. In the following code, the k-means algorithm is executed for 5 clusters (climatic regions). As the algorithm is sensitive to the initialization of the cluster centers, the best of 10 different initializations is chosen.

```
kmeans = KMeans(
    n_clusters=5, n_init=10,
    metric=l2_distance)
clusters = kmeans.fit_predict(X)
```

In Figure 2 we can see each weather station in the map of Spain, colored according to their cluster. The resulting distribution fits very well with climatic maps of Spain. The red points, located only in the Canary Islands, would correspond to the subtropical climate. The green points, in the north of mainland Spain, could represent the Atlantic climate. Yellow stations are mainly distributed on the Mediterranean coast suggesting the so-called Mediterranean climate. Orange points cover the interior of mainland Spain, which has a continental climate. Finally, the purple stations are scattered on the coldest



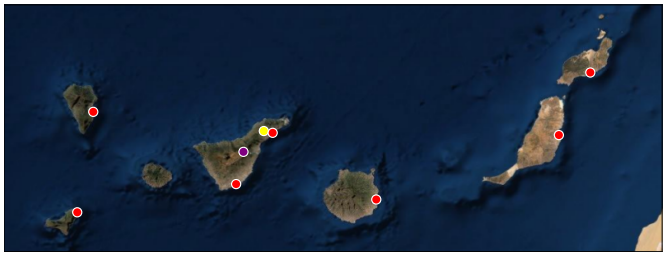


Fig. 2. Weather stations clusters based on temperature curves in mainland Spain (top) and on the Canary Islands (bottom).

points of Spain, including some mountain ranges, and are thus examples of cold or high mountain climates.

Focusing on the Canary Islands, we can observe two points (yellow-mediterranean and purple-mountain) that, at a first glance, seem to be mislabeled. A closer inspection, however, shows that the "Mediterranean" station is the airport of Los Rodeos, characterized for having dense fog and a lower temperature than their surroundings [19]. The cold-mountain station corresponds to an observatory located on Mount Teide (the highest mountain in Spain) at 2390 meters high.

Although the clustering of the complete curves seems really accurate, it is difficult to interpret how the labels have been assigned. Dimensionality reduction techniques may provide some interpretability as they allow us to identify a few characteristics of interest. The most popular dimensionality reduction method is, probably, principal component analysis (PCA), which projects the data in the (orthogonal) directions of maximal variance. In the functional principal component analysis (FPCA), the idea remains the same, but calculations need to be adapted. As an example, projections are made with the L^2 inner product $\langle x_1, x_2 \rangle_{L^2} = \int_{\mathcal{T}} x_1(t)x_2(t)dt$. The following code obtains the first two principal components (`fit`) and projects the data on these directions (`transform`).

```
fpca = FPCA(n_components=2)
fpca.fit(X)
X_red = fpca.transform(X)
```

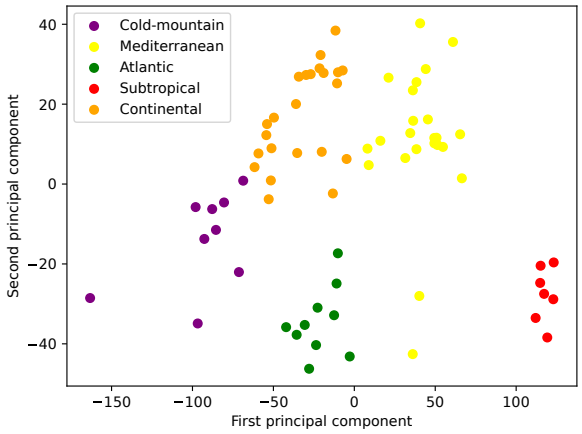


Fig. 3. AEMET data projected to their first two principal components.

The first component captures the average temperature. The second one marks the difference between summer (maximum of temperatures) and winter (minimum). In some sense, it measures the thermal amplitude. Moreover, dimensionality reduction allow us to visualize the data in the reduced space. For example, Figure 3 shows a scatter plot of the temperatures in the space of the first two principal components. Colors are those of the clusters obtained previously with k-means. Following the previous discussion, we can interpret the first component (x axis) as the average temperature, from colder (left) to warmer (right) locations. On the other hand, the second component (y axis) represents the thermal amplitude, from small variations (bottom) to higher differences (top) between summer and winter. We can see that stations from Canary Islands (red points) are quite different to the others (as seen in Figure 1), with warmer average temperatures and low variation, which are characteristics of the subtropical climate. Green points have colder temperatures, but have low thermal amplitude too. These points correspond to the northern coast of mainland Spain, having an Atlantic climate. Continental and Mediterranean climates (orange and yellow) present a much more pronounced amplitude, with the difference that continental climate has lower overall temperatures. However, two stations identified as Mediterranean by k-means seem to be mislabeled in this PCA representation. With a closer inspection, we see that one of these outliers is, precisely, the already mentioned airport of Los Rodeos, which has "Mediterranean" average temperature, with the low annual variations of the Canary Islands, where it is located. The other corresponds to Tarifa, in the Strait of Gibraltar. It is a very windy place which receives the cold water from the Atlantic. This causes Tarifa to have its own micro-climate characterized by smoother annual temperatures, as shown in the scatter plot. Finally, purple locations (cold-mountain climate) exhibit a variety of amplitudes, but are characterized for the lowest average temperatures. In particular, the leftmost point corresponds to Navacerrada station, the magnitude outlier detected in Figure 1.

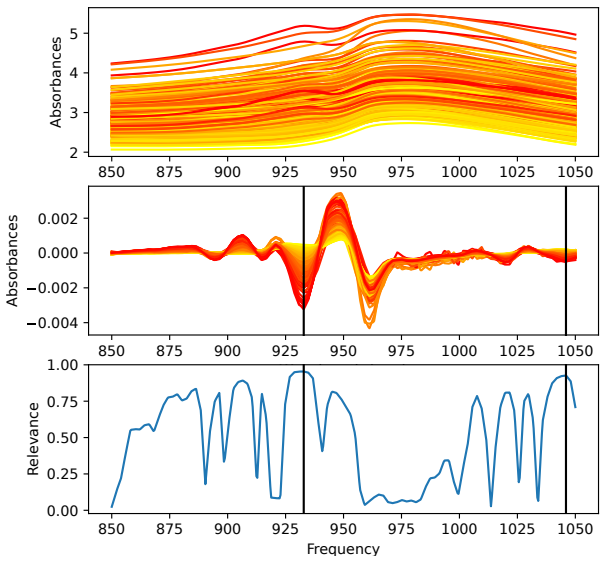


Fig. 4. On top, the curves of the Tecator dataset. A color gradient indicates the fat content. The middle plot shows the derivative of these curves. The bottom plot shows the dependency of the target on each function point. The two points of higher dependency have been marked with vertical black lines.

IV. Spectrometric data: derivatives, regression, and variable selection

In this section we use the Tecator dataset [20] to illustrate the regression problem and some new functionalities. The data consists of $N = 215$ observations of $M = 100$ channel spectra of absorbances of different pieces of meat, as well as their moisture, fat and protein contents. In this case, response variables are retrieved jointly with the functional data. We only keep the fat content as target for regression [11]. Trajectories are plotted in the upper part of Figure 4 using a color gradient for the fat content.

```
X, y = fetch_tecator(return_X_y=True)
y = y[:, 0]
X.plot(gradient_criteria=y)
```

We can appreciate that the magnitude of each curve bears almost no correlation with its fat contents. Nevertheless, it is well known in the literature that this problem becomes easier by using the second derivative of the trajectories [11]. This can be a convenient preprocessing which is exclusive of functional data. So, we compute the second derivative of the data:

```
X_der = X.derivative(order=2)
```

The resulting derivatives are in the middle plot of Figure 4. After derivation, the fat content is clearly proportional to the magnitude of the curves at several points.

Before performing regression, as explained in [1], a common approach to prevent overfitting when working with continuous regression coefficients is to use the representation in a basis expansion. As an example, we represent the original data in a B-spline basis with $B = 10$ elements with *scikit-fda*:

```
basis = BSpline(n_basis=10)
X_der_basis = X_der.to_basis(basis)
```

We can split the data in train and test partitions using, for example, the `train_test_split` function from *scikit-learn*, which is able to deal with the functional objects of *scikit-fda*. Then we use the standard functional linear regression model with scalar response $y = \beta_0 + \int_{\mathcal{T}} \beta_1(t)x(t)dt$ [1]. The following code fits the regression model, obtains the predicted values for the test partition, and compute the score of the prediction in terms of the coefficient of determination R^2:

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
score = r2_score(y_test, y_pred)
```

Some alternatives available in the library include nonparametric models based on nearest neighbors or kernel regression It is even possible to add additional regularization terms for the coefficients, if needed. Nevertheless, even with this simple model, we are able to obtain a R^2 score of 0.951 by using the second derivatives of the Tecator trajectories.

Although this model has a good performance, it lacks interpretability. In the plot of the derivatives, it is easy to identify a couple of points that should provide a good prediction by themselves. Thus, we would expect that a variable selection procedure could find those points of interest and reduce the dimensionality of the data obtaining a more interpretable model. Many classical multivariate methods for variable selection do not work with continuous functional data due to the high redundancy between close variables. The package *scikit-fda* offers several variable selection methods that prevent this problem, from adaptations of multivariate methods that take redundancies into account, such as mRMR [21], to purely functional methods that consider the nature of these data, such as a proposal based on reproducing kernel Hilbert spaces [22] or recursive maxima hunting [23]. Here, we use the maxima hunting method (MH), that computes the relevance of each point $x(t)$ as its relation with the response variable (quantified with a measure of statistical dependence). MH selects the local maxima of the resultant relevance function, what automatically removes redundant variables [8]. In the following example we use MH to select the two most relevant local maxima.

```
var_sel = MaximaHunting(
    local_maxima_selector=(
        RelativeLocalMaximaSelector(
            max_points=2)))
X_mv = var_sel.fit_transform(X_der, y)
```

In the bottom plot of Figure 4 we can see the relevance function for Tecator derivatives with the distance correlation measure [24], and the two selected variables. These are also marked over the derivative curves in the middle plot, where we can appreciate that they correspond to points where the fat content was proportional to the magnitude of the curves.

After selection we can apply any multivariate regression model from the *scikit-learn* library. For example, the standard linear model obtains a R^2 score of 0.917. This is slightly worse than the score obtained with the whole trajectories, but

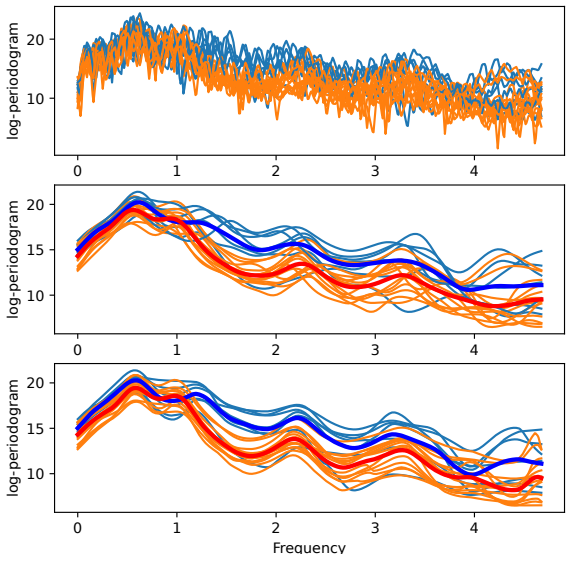


Fig. 5. Ten curves from the Phoneme dataset with classes "aa" (in blue) and "ao" (in orange). We can see original curves (top), smoothed curves (middle), and the curves after per-class registration (bottom). Class means are plotted in thick darker lines in the last two plots.

it is using only two variables, what entails a significant gain in interpretability. Moreover, we could even use regression trees or any other method available in the *scikit-learn* to try to improve the performance without losing interpretability.

V. Voice signals: smoothing, registration, and classification

The Phoneme dataset [25] was chosen to illustrate smoothing, functional data registration (alignment) and classification. This dataset contains curves corresponding to 4509 log-periodograms with the pronunciation of five different phonemes by 50 male speakers: "sh" as in "she", "dcl" as in "dark", "iy" as the vowel in "she", "aa" as the vowel in "dark", and "ao" as the first vowel in "water". Here, we focus on the binary classification problem, so we only consider the $N = 1717$ curves of the first two classes, "aa" (class 0) and "ao" (class 1), which are the most difficult to distinguish. The domain of the functions is also restricted to the first 150 variables as suggested in [11].

Analogously to previous examples, we download the data using the `fetch_phoneme` function. The top plot in Figure 5 shows the first 20 curves with a different color per class.

```
X[:20].plot(group=y)
```

We can observe that phoneme trajectories are particularly noisy, which may complicate further analysis. This can be solved by smoothing the curves, for example, using a weighted average of neighbouring points $\hat{x}(t) = \int_{\mathcal{T}} w_t(s)x(s)ds$, where $\hat{x}$ is a smoothing estimation of the underlying signal and $w_t(s)$ has its maximum at $s = t$ and decreases monotonically from that point. Several smoothing strategies are available in *scikit-fda* ranging from different kernel smoothers, widely used in density estimation, to smoothing via representation in a basis, which also allows penalizing the curvature to achieve additional smoothness. Here, we smooth the Phoneme observations with a Nadaraya-Watson kernel smoother. The bandwith parameter controls the degree of smoothing, and has to be adjusted to prevent infra or over-smoothing.

```
smoother = KernelSmoother(
    NadarayaWatsonHatMatrix(
        bandwidth=0.1, kernel=normal)))
X_smooth = smoother.fit_transform(X)
```

The first 20 smoothed curves are shown in the middle plot of Figure 5. We can see that the per-class means has much less pronounced maxima and minima than the individual observations. This can be an indicator of misalignment. Misalignment is a new challenge that did not appear in multivariate statistics. It affects not only to the mean estimation, but methods such as variable selection, that assume aligned data to start with. Functional data registration is the process for which the data is aligned so that maxima, minima, or other relevant landmarks appear at the same points in each observation [7]. The package *scikit-fda* offers shifting and elastic registration procedures. In this case we do not have information about the landmarks of interest, so we can use an advanced elastic registration procedure based on the properties of the Fisher-Rao distance [26]. For example, the following code registers together all curves from the first class:

```
reg = FisherRaoElasticRegistration(
    penalty=0.01)
X_reg = reg.fit_transform(X_smooth[y==0])
```

The result of the per-class registration is displayed in the bottom plot of Figure 5. We can see that the registered curves have now their landmarks properly aligned, and the class means more closely resemble an typical trajectory of the corresponding class. More importantly, we can appreciate more clearly that the disposition of maxima and minima is different between classes. Hence, we should not attempt to register all curves together.

As we have just seen, aligning all the curves at the same time entails a loss of discriminant information. Therefore, for the classification task, we consider the unaligned smoothed functions of the middle plot in Figure 5. Moreover, a classifier that is resilient to unaligned data should be preferable. The package *scikit-fda* offers a variety of classification algorithms for functional data: distance-based classifiers such as nearest centroids or k-nearest neighbors; depth-based classifiers, including maximum-depth [27] and the DD^{G} classifier [28]; or even functional logistic regression [29]. As an example, we use a k-nearest neighbors classifier with the functional Mahalanobis distance proposed in [30].

```
classifier = KNeighborsClassifier(
    n_neighbors=67,
    metric=MahalanobisDistance())
```

The sample is split in train and test, leaving the 30% of the data as the test partition. We then fit the model and compute the predictions, obtaining an accuracy score of 0.805:

```
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
score = accuracy_score(y_test, y_pred)
```

For simplicity, we have fixed the number of neighbors to $\sqrt{N}$. In a complete analysis is often better to choose the values for the number of neighbors and the smoothing parameter via a cross validation procedure. As most objects in *scikit-fda* have the same application programming interface (API) as *scikit-learn* classes [31], the hyperparameter selection and cross validation utilities of that package can be directly applied, making this an easy task.

VI. Conclusions

We have presented three real-data examples to illustrate the functionalities of the Python *scikit-fda* package. As seen from these use cases, *scikit-fda* offers powerful functional data methods, easily integrable with the multivariate tools available in *scikit-learn*. In addition to the showcased methods for preprocessing, exploratory analysis and machine learning, *scikit-fda* offers a variety of additional tools. These functionalities include generation of synthetic data, feature construction, interactive visualization, interpolation, extrapolation, regularization or statistical inference. They are thoroughly described and illustrated with examples in the official documentation of the package. All of the provided methods can be employed in a variety of real applications, which makes FDA, and in particular the package *scikit-fda*, a useful new tool for data analysts, machine learning researchers and practitioners.

Acknowledgment

The authors acknowledge financial support from the Spanish Ministry of Education and Innovation, projects PID2019-106827GB-I00 / AEI / 10.13039/501100011033 and PID2019-109387GB-I00, and from grant FPU18/00047.

References

[1] J. Ramsay, B. W. Silverman, Functional Data Analysis, 2nd Edition, Springer Series in Statistics, Springer-Verlag, New York, 2005.

[2] D. Ballari, R. Giraldo, L. Campozano, E. Samaniego, Spatial functional data analysis for regionalizing precipitation seasonality and intensity in a sparsely monitored region: Unveiling the spatio-temporal dependencies of precipitation in Ecuador, International Journal of Climatology 38 (8) (2018) 3337–3354.

[3] M. Pouplier, J. Cederbaum, P. Hoole, S. Marin, S. Greven, Mixed modeling for irregularly sampled and correlated functional data: Speech science applications, The Journal of the Acoustical Society of America 142 (2) (2017) 935–946.

[4] A. Pourshoghi, I. Zakeri, K. Pourrezaei, Application of functional data analysis in classification and clustering of functional near-infrared spectroscopy signal in response to noxious stimuli, Journal of Biomedical Optics 21 (10) (2016) 101411.

[5] T. Boschi, J. Di Iorio, L. Testa, M. A. Cremona, F. Chiaromonte, Functional data analysis characterizes the shapes of the first COVID-19 epidemic wave in Italy, Scientific Reports 11 (1) (2021) 17054.

[6] A. Delaigle, P. Hall, Defining probability density for a distribution of random functions, The Annals of Statistics 38 (2) (2010) 1171–1193.

[7] J. S. Marron, J. O. Ramsay, L. M. Sangalli, A. Srivastava, Functional data analysis of amplitude and phase variation, Statistical Science 30 (4) (2015) 468–484.

[8] J. R. Berrendero, A. Cuevas, J. L. Torrecilla, Variable selection in functional data classification: A maxima-hunting proposal, Statistica Sinica 26 (2) (2016) 619–638.

[9] C. Ramos-Carreño, A. Suárez, J. L. Torrecilla, M. Carbajo Berrocal, P. Marcos Manchón, P. Pérez Manso, A. Hernando Bernabé, D. García Fernández, Y. Hong, P. M. Rodríguez-Ponga Eyriès, Á. Sánchez Romero, E. Petrunina, Á. Castillo, D. Serna, R. Hidalgo, scikit-fda: Functional data analysis in python (2022).
URL https://github.com/GAA-UAM/scikit-fda

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[11] F. Ferraty, P. Vieu, Nonparametric Functional Data Analysis: Theory and Practice, Springer Series in Statistics, Springer-Verlag, New York, 2006.

[12] D. Díaz-Vico, C. Ramos-Carreño, scikit-datasets: Scikit-learn-compatible datasets (Mar. 2022).
URL https://github.com/daviddiazvico/scikit-datasets

[13] Y. Sun, M. G. Genton, Functional boxplots, Journal of Computational and Graphical Statistics 20 (2) (2011) 316–334.

[14] I. Gijbels, S. Nagy, On a general definition of depth for functional data, Statistical Science 32 (4) (2017) 630–639.

[15] R. Fraiman, G. Muniz, Trimmed means for functional data, Test 10 (2) (2001) 419–440.

[16] S. López-Pintado, J. Romo, On the concept of depth for functional data, Journal of the American Statistical Association 104 (486) (2009) 718–734.

[17] W. Dai, M. G. Genton, Multivariate functional data visualization and outlier detection, Journal of Computational and Graphical Statistics 27 (4) (2018) 923–934.

[18] A. Arribas-Gil, J. Romo, Shape outlier detection and visualization for functional data: The outliergram, Biostatistics 15 (4) (2014) 603–619.

[19] V. M. Romeo, M. V. Marzol Jaén, Análisis del viento y la niebla en el aeropuerto de Los Rodeos (Tenerife). Cambios y tendencias, in: S. Fernández Montes, F. Sánchez Rodrigo (Eds.), Cambio climático y cambio global., no. 9 in Publicaciones de la Asociación Española de Climatología. Serie A., Asociación Española de Climatología, 2014, pp. 325–334.

[20] C. Borggaard, H. H. Thodberg, Optimal minimal neural interpretation of spectra, Analytical Chemistry 64 (5) (1992) 545–551.

[21] J. R. Berrendero, A. Cuevas, J. L. Torrecilla, The mRMR variable selection method: A comparative study for functional data, Journal of Statistical Computation and Simulation 86 (5) (2016) 891–907.

[22] J. R. Berrendero, A. Cuevas, J. L. Torrecilla, On the use of reproducing kernel Hilbert spaces in functional classification, Journal of the American Statistical Association 113 (523) (2018) 1210–1218.

[23] J. L. Torrecilla, A. Suárez, Feature selection in functional data classification with recursive maxima hunting, in: Advances in Neural Information Processing Systems 29, Curran Associates, Inc., 2016, pp. 4835–4843.

[24] G. J. Székely, M. L. Rizzo, N. K. Bakirov, Measuring and testing dependence by correlation of distances, The Annals of Statistics 35 (6) (2007) 2769–2794.

[25] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition, Springer Series in Statistics, Springer-Verlag, New York, 2009.

[26] A. Srivastava, W. Wu, S. Kurtek, E. Klassen, J. S. Marron, Registration of functional data using Fisher-Rao metric (May 2011). doi:10.48550/arXiv.1103.3817.

[27] A. Cuevas, M. Febrero, R. Fraiman, Robust estimation and classification for functional data via projection-based depth notions, Computational Statistics 22 (3) (2007) 481–496.

[28] J. A. Cuesta-Albertos, M. Febrero-Bande, M. Oviedo de la Fuente, The DD^G-classifier in the functional setting, TEST 26 (1) (2017) 119–142.

[29] B. Bueno-Larraz, J. R. Berrendero, A. Cuevas, On functional logistic regression: Some conceptual issues (Jul. 2021). doi:10.48550/arXiv.1812.00721.

[30] J. R. Berrendero, B. Bueno-Larraz, A. Cuevas, On Mahalanobis distance in functional settings, Journal of Machine Learning Research 21 (9) (2020) 1–33.

[31] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: Experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.