Original Article

Healthc Inform Res. 2022 July;28(3):210-221.
https://doi.org/10.4258/hir.2022.28.3.210
pISSN 2093-3681 • eISSN 2093-369X

HIR
Healthcare Informatics Research

Machine Learning Smart System for Parkinson Disease Classification Using the Voice as a Biomarker

Ilias Tougui, Abdelilah Jilbab, Jamal El Mhamdi
E2SN, ENSIAS, Mohammed V University in Rabat, Rabat, Morocco

Objectives: This study presents PD Predict, a machine learning system for Parkinson disease classification using voice as a biomarker. **Methods:** We first created an original set of recordings from the mPower study, and then extracted several audio features, such as mel-frequency cepstral coefficient (MFCC) components and other classical speech features, using a windowing procedure. The generated dataset was then divided into training and holdout sets. The training set was used to train two machine learning pipelines, and their performance was estimated using a nested subject-wise cross-validation approach. The holdout set was used to assess the generalizability of the pipelines for unseen data. The final pipelines were implemented in PD Predict and accessed through a prediction endpoint developed using the Django REST Framework. PD Predict is a two-component system: a desktop application that records audio recordings, extracts audio features, and makes predictions; and a server-side web application that implements the machine learning pipelines and processes incoming requests with the extracted audio features to make predictions. Our system is deployed and accessible via the following link: https://pdpredict.herokuapp.com/. **Results:** Both machine learning pipelines showed moderate performance, between 65% and 75% using the nested subject-wise cross-validation approach. Furthermore, they generalized well to unseen data and they did not overfit the training set. **Conclusions:** The architecture of PD Predict is clear, and the performance of the implemented machine learning pipelines is promising and confirms the usability of smartphone microphones for capturing digital biomarkers of disease.

Keywords: Parkinson Disease, Voice Disorders, Machine Learning, Diagnosis, Computer-Assisted, Medical Informatics Applications

Submitted: January 19, 2022
Revised: 1st, April 24, 2022; 2nd, June 2, 2022
Accepted: June 28, 2022

Corresponding Author
Ilias Tougui
E2SN, ENSIAS, Mohammed V University in Rabat, United Nations Avenue, Agdal, Rabat, Morocco. Tel: +212638409439, E-mail: ilias_tougui@um5.ac.ma (https://orcid.org/0000-0001-7790-4284)

I. Introduction

Parkinson disease (PD) is a progressive neurodegenerative disorder that affects older individuals [1]. PD patients experience various motor symptoms, including bradykinesia, rigidity, and tremor, and non-motor symptoms such as memory disturbances and sleep disorder [1]. It is primarily treated by increasing dopamine levels using pharmacological therapy or surgery [2]. Research has shown that the voice is an early biomarker of this disease. The gradual deterioration of communication skills and speech impairment are common symptoms in most patients. Ho et al. [3] found that 147 of 200 PD patients showed deterioration in speech

characteristics. Thus, early detection of PD is essential for disease management. Assessing the voice as a biomarker of the disease could be critical in improving PD diagnosis to prevent patients from developing unmanageable motor complications that may decrease quality of life.

The traditional PD diagnosis is primarily clinical; it is expensive and can take hours to a few days to perform. PD is usually diagnosed by a neurologist based on the patient's medical history, a review of signs and symptoms, and a neurological and physical examination. This diagnostic process makes it difficult for patients to go to the hospital in person, especially in the mild stages of the disease. However, with advances in technology and computational capabilities, machine learning (ML) has emerged as a valuable tool for the early prediction of diseases. With high-quality data, which are usually recorded using professional equipment or wearable sensors and mobile devices, ML applications have the potential to assist doctors in the diagnostic process and clinical decision-making. Research has shown that mobile sensors embedded in smartphones, such as accelerometers, gyroscopes, and microphones, can help researchers develop valuable tools by effectively detecting digital biomarkers of diseases [4].

While most studies have focused only on research [5-10], we focused on both research and development. To this end, we present PD Predict: an intelligent system capable of predicting PD using ML and voice as a biomarker. PD Predict consists of two main components: a client-side desktop application that records audio data, extracts audio features, and makes predictions; and a server-side web application that implements ML models and predicts PD using the extracted audio features. In this paper, we first describe the creation of our original dataset from smartphone recordings drawn from the mPower database [11]. Next, we describe in detail the creation of the ML pipelines, and finally, we present the architecture of our proposed system. This paper is organized as follows: Section II describes our methodology in detail, Section III presents the results, and Section IV discusses the findings.

II. Methods

1. Dataset Creation

1) Data acquisition

This study's database is based on the mPower Parkinson study database obtained from the mPower Public Research Portal [11,12]. mPower is a Parkinson's disease clinical study performed only through a mobile application interface that consists of seven tasks that each participant must complete, three survey questionnaires (a demographic survey, the Unified-Parkinson's Disease Rating Scale survey, and the Parkinson's Disease-Questionnaire-8), and four activities (the memory task, the tapping task, the voice task, and the walking task). In this study, we were only interested in two tasks (the voice activity task and the demographic survey). Our original dataset was created in three stages: acquiring raw audio recordings, selecting valid participants, and finally extracting audio features. Python and SQL were used with the Synapse client to collect the voice recordings [13]. The demographic survey was then used to differentiate between participants with Parkinson's disease (PWPs) and healthy controls (HCs).

2) Participant selection

Phase 1: Participant selection using the demographic survey

Participants were categorized as PWPs if they were diagnosed professionally by a physician, with a valid age and date of diagnosis. It was specified that they were parkinsonians (not caregivers) and had never undergone surgery or deep brain stimulation. Participants who were not formally diagnosed by a physician, had a valid age and a valid date of diagnosis, and had no movement symptoms were considered HCs.

Phase 2: Participant selection using the recordings' medication-associated time points

Participants were requested to record their voice three times a day in the voice activity task, saying "Aaah" for 10 seconds at a steady pace using the smartphone's microphone. HCs could record their voice whenever they wanted. In contrast, PWPs were requested to record their voice if they took PD medication at three specific times: before taking PD medication, after taking PD medication, and at another time. Otherwise, they were allowed to record their voices three times a day, whenever they wanted. In this phase, we only selected the recordings of participants who did not take PD medication or who recorded their voices before taking PD medication. Each participant had a unique health-code identifier, which was used in the next phase.

Phase 3: Participant selection to perform an equal case-control study

To maximize efficiency in this case-control study, we evenly distributed the participants between the two groups (PWPs and HCs). In the original database, some participants recorded their voice many times, while others recorded their voice only once. In this step we selected only two recordings from each participant, that recorded their voice more than once. Next, we examined each recording to verify that we

selected valid recordings with minimal background noise. We also matched the groups' participants by age and sex. We specifically selected participants who were 40 years of age and older, as PD predominantly affects older individuals [14]. The final cohort chosen is described in Table 1.

3) Audio feature extraction

Feature extraction is critical in ML and pattern recognition systems [15], mainly when dealing with audio data. Because audio signals are non-stationary, feature extraction is performed on a frame basis by dividing the signals into short frames [16]. All the recordings were divided into 25-ms windows with a 10-ms step size (usually between 20 and 40 ms [16]) and sampled at 44.1 kHz. Using the Surfboard [17] library, we extracted 123 audio features. This Python library is built with state-of-the-art audio analysis packages and provides an easy-to-use API to extract several audio features applicable in the medical domain. Surfboard allows the calculation of important audio feature statistics to obtain a feature-rich dataset. Table 2 summarizes the extracted features and showcases the structure of our final dataset.

4) Dataset partitioning

Before starting the modeling phase, we divided the dataset into two sets: a training set and a holdout set. The training set was used to train the pipelines and evaluate their performance using subject-wise cross-validation (CV), while the holdout set was used to verify the models' generalizability

Table 1. Final distribution of valid subjects in this study

	PD group	HC group	Total
Number of recordings	424	424	848
Number of participants	212	212	424
Sex			
Male	161	161	322
Female	51	51	102
Age (yr)	58.97 ± 8.95 (40–79)	58.97 ± 8.95 (40–79)	

Values are presented as mean ± standard deviation (min-max).
PD: Parkinson disease, HC: healthy controls.

Table 2. Extracted features and structure of the dataset

Feature Id	Feature/Component	Feature statistics
1–78	MFCCs	
79–84	F0 Contour	
85–86	F0	
87–92	Intensity	Mean
93	Log energy	Std
94–99	Sliding-window Log energy	
100	Loudness	First derivative mean
101	Pitch period entropy	First derivative std
102–106	Jitters	
107–111	Shimmers	Second derivative mean
112	Detrended fluctuation analysis	Second derivative std
113–116	Formants	
117	HNR	
118–123	RMS	
124	Class (PWP = 1, HC = 0)	

MFCC: mel-frequency cepstral coefficient, HNR: harmonic to noise ratio, RMS: root mean square, PWP: participants with Parkinson disease, HC: healthy controls.

 https://doi.org/10.4258/hir.2022.28.3.210

Table 3. Distribution of participants in the training and holdout sets

	Training set (80%)		Holdout set (20%)		Total
	PD group	HC group	PD group	HC group	
Number of recordings	340	340	84	84	848
Number of participants	170	170	42	42	424
Sex					
Male	126	131	35	30	322
Female	44	39	7	12	102
Age (yr)	59.13 ± 9.30 (40–79)	58.81 ± 8.97 (40–79)	58.31 ± 7.42 (43–75)	59.62 ± 8.90 (43–76)	

Values are presented as mean ± standard deviation (min–max).
PD: Parkinson disease, HC: healthy controls.

Table 4. Hyperparameters of the two pipelines

Pipeline	Stage	Hyperparameter	Value
gbcpl	Imputation	strategy	mean
	Standardization	with_mean	true
		with_std	true
	Feature selector	estimator = lasso	
		alpha	0.001
		tol	0.1
		max_features	60
	GBC classifier	n_estimators	600
		min_samples_split	0.8
		min_samples_leaf	0.5
		max_features	52
		max_depth	8.0
		learning_rate	0.1
gbcpen	Imputation	strategy	mean
	Standardization	with_mean	true
		with_std	true
	Feature selector	estimator = elasticnet	
		tol	3.0004
		max_iter	1000000
		l1_ratio	0.02
		alpha	0.52
		max_features	60
	GBC classifier	n_estimators	200
		min_samples_split	0.70001
		min_samples_leaf	0.30004
		max_features	26
		max_depth	11.0
		learning_rate	0.04

GBC: gradient boosting classifier.

on unseen data. For data partitioning and using the subject-wise division approach [18], we allocated 80% of the dataset to the training set and 20% to the holdout set. This division guarantees that the subjects (or participants) and their relative recordings in the training set are independent of the subjects and their relative recordings in the holdout set. Table 3 details the distribution of participants in both sets.

2. Classification Pipelines

1) Data preprocessing

In this step, we applied two forms of preprocessing [19] to the dataset: data imputation and data scaling. Data imputation replaces missing values in a feature vector with the mean, the median, or the most frequent value of the same feature vector; this step is required because missing values can cause problems for ML algorithms. Data scaling standardizes the dataset, as it contains various features with varying scales. This step ensures that all features contribute equally to the learning process. Table 4 presents more details.

2) Feature selection

Feature selection is an essential concept in ML. Having a proper subset of features reduces the complexity of a model, enables it to be trained faster, and makes it easier to interpret. Researchers use various feature selection techniques [20], such as filter, wrapper, and embedded methods. In this work, we used two embedded methods. The best subset of features was selected using a meta-transformer based on feature importance, with Lasso or ElasticNet [21] as the primary estimator.

3) Machine learning pipelines

To develop our models, we used pipelines. A pipeline automates an ML workflow by dividing it into reusable independent modular parts: in our case, data imputation, data scaling, feature selection, and then classification. In this study, we created two pipelines: a gradient boosting classifier (GBC) pipeline with Lasso (gbcpl) and a GBC pipeline with ElasticNet (gbcpen); their hyperparameters were tuned using the randomized search technique [22] (Table 3). To cor-

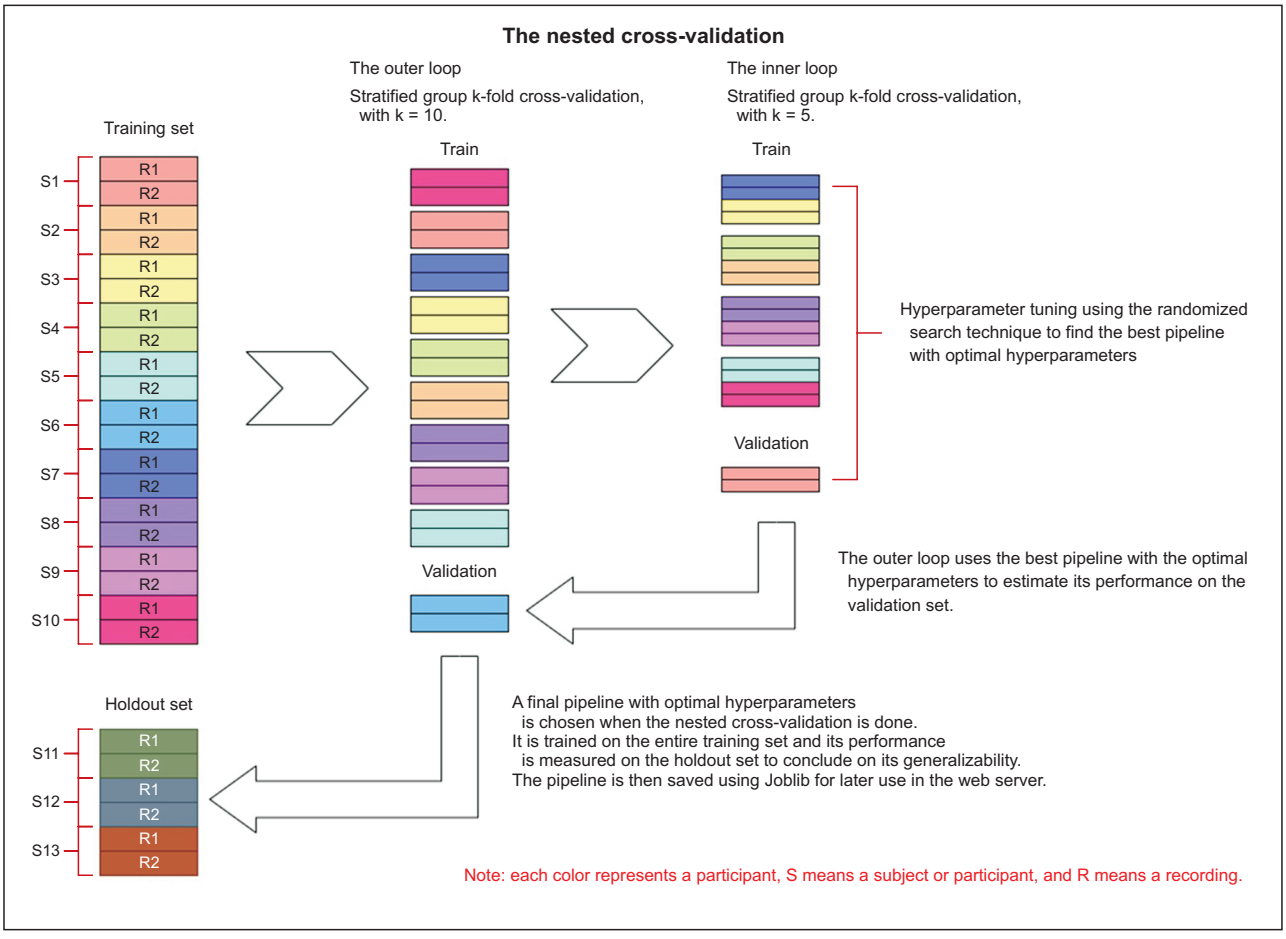

Figure 1. Schematic illustration of the nested cross-validation approach.

 https://doi.org/10.4258/hir.2022.28.3.210

rectly estimate the performance of the pipelines, we used a nested CV approach, where two subject-wise CV techniques were grouped to form two loops, an inner loop and an outer loop, as shown in Figure 1. In the outer loop, we used the stratified-group k-fold CV technique with k = 10. In each loop, the training set is divided into 10 parts or folds; nine parts are used to train the pipelines, and the remaining part is used for validation. The inner loop divides the nine parts into five using stratified-group k-fold CV with k = 5. Four parts are used to fit several pipelines with random hyperparameters. The remaining part is used to select the best pipeline with specific hyperparameters by maximizing a scoring metric (in our case, recall). When the best pipeline is found in the inner loop, it is used by the outer loop to estimate its performance on the validation set. This step is repeated 10 times, and the performance of the pipelines is reported as the mean (standard deviation) for each performance metric. This technique is time-consuming, considering the time needed to find optimal hyperparameters by the inner loop. However, it has been demonstrated to produce accurate results and reduces overfitting when dealing with small datasets [23]. To assess the performance of the pipelines, we calculated three performance measures: accuracy, recall, and the F1-score.

3. Implementation

As previously described, this work is divided into two parts: the dataset creation part (from data acquisition and par-

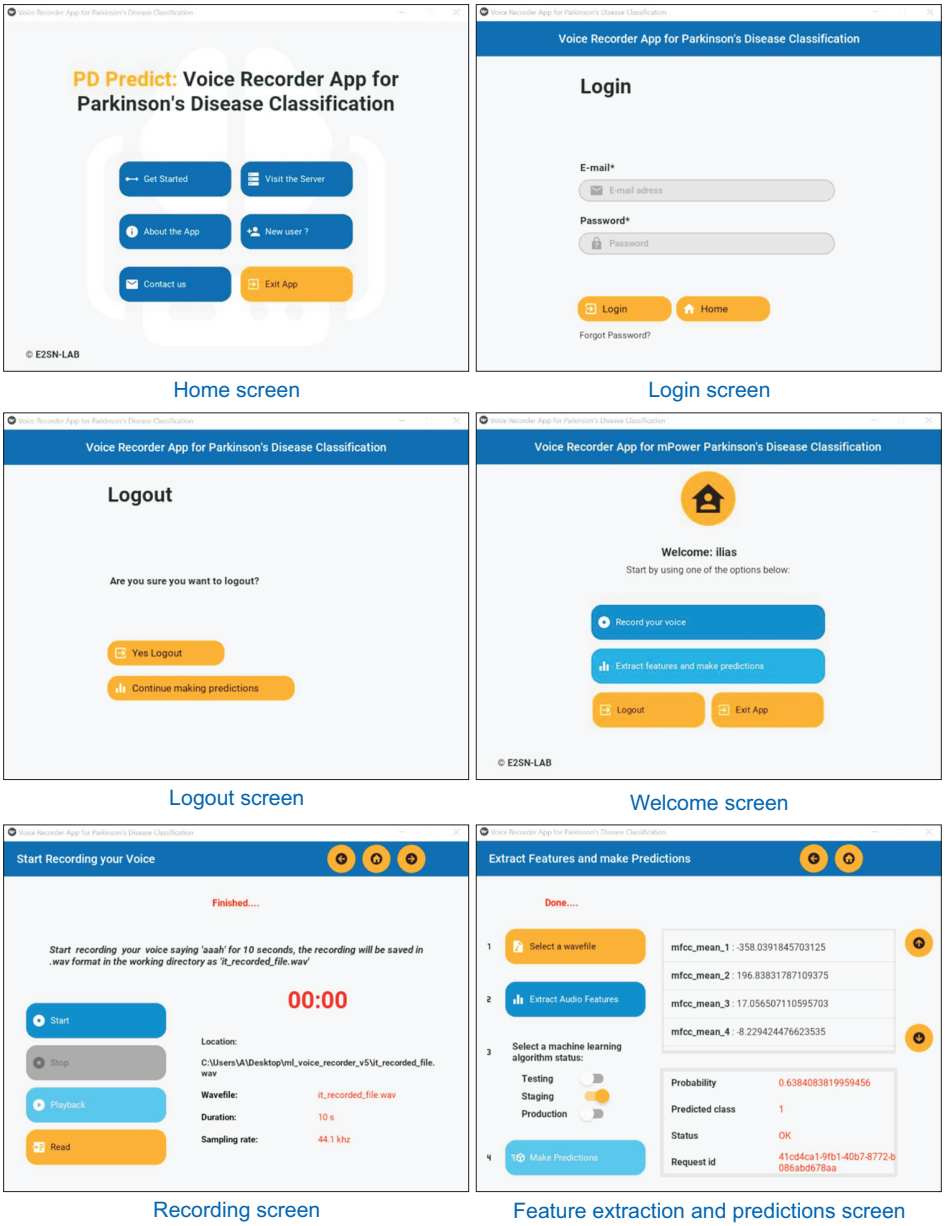

Figure 2. Various screens in the client-side desktop application.

ticipant filtering to audio feature extraction and data partitioning) and the pipeline creation part (data preprocessing, feature selection, and tuning the selected ML pipelines). To implement those two parts, we developed PD Predict. PD Predict is an intelligent system that classifies PD using ML and voice. It can be divided into two main components: a client-side desktop application that allows recording voice data, extracting audio features, and making predictions; and a server-side web application that implements our ML pipelines and allows user management, database indexing, and processing incoming requests with audio features to classify PD.

1) Client-side desktop application

The client-side desktop application is a Windows application developed entirely using Python and Kivy [24], an open-source, cross-platform graphical framework for natural user interface development. Our application has six screens with three main features: recording audio data, extracting audio features and communicating with the server via HTTP requests (Figures 2, 3A). Voice recording is done using PyAudio [25], which provides Python bindings for PortAudio, a cross-platform audio I/O library used to record and play audio on desktop operating systems such as Windows, Linux, and macOS. Each recorded file is sampled at 44.1 kHz and saved in .wav format with a 10-second duration. Feature extraction is performed using Surfboard; we can extract audio features either from the recorded .wav file or from a test .wav file that is available on the computer (for instance, using test recordings from the holdout set). Extracted features are then sent to the server via HTTP requests in the JSON format to a prediction endpoint. This endpoint is developed using a REST API, where each HTTP request includes a header with user credentials in the form of a token that is generated for the user when successfully logged in to the server. Only authen-

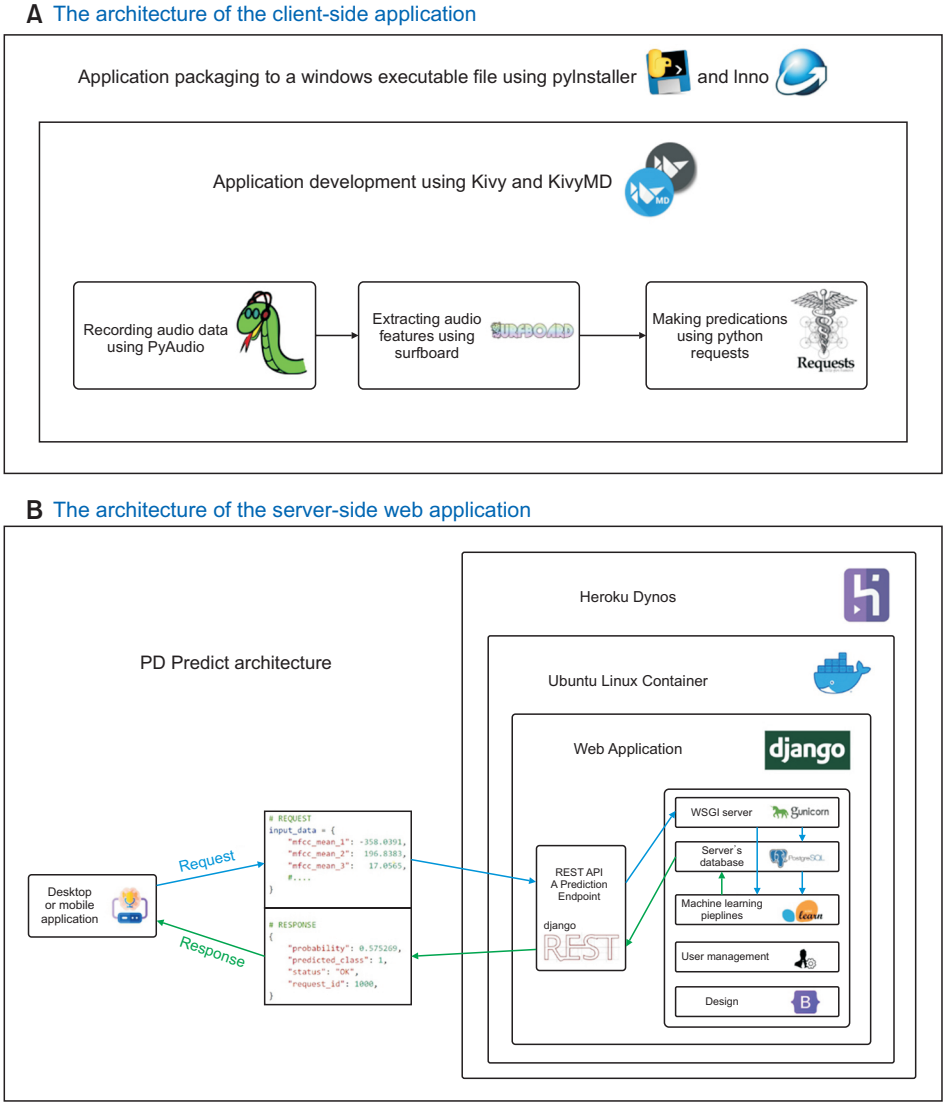

Figure 3. Architecture of PD Predict: (A) client-side application and (B) server-side web application.

https://doi.org/10.4258/hir.2022.28.3.210

ticated users to the server can use the desktop application. An HTTP response is returned to the application in JSON format when the server successfully authorizes the user and finishes processing the request with the extracted features by a selected ML pipeline. We packaged our application into a Windows executable file using PyInstaller [26] and Inno [27] for ease of use.

2) Server-side web application

The server-side web application is developed entirely using Django [28], an open-source, high-level Python web framework that encourages rapid development and clean, pragmatic design. The server's database is based on PostgreSQL, an open-source relational database management system. The ML pipelines were trained using Scikit-learn [29], saved using Joblib, and accessed via a prediction endpoint developed using Django Rest Framework. When a user posts a request to the server with the extracted audio features, those features are deserialized by the REST API and stored in the database to be processed by the selected ML pipeline. When processing is done, the response of the ML pipeline is stored in the database and then serialized by the REST API in a JSON format to be used by the desktop application. The web application also provides a mechanism for user management

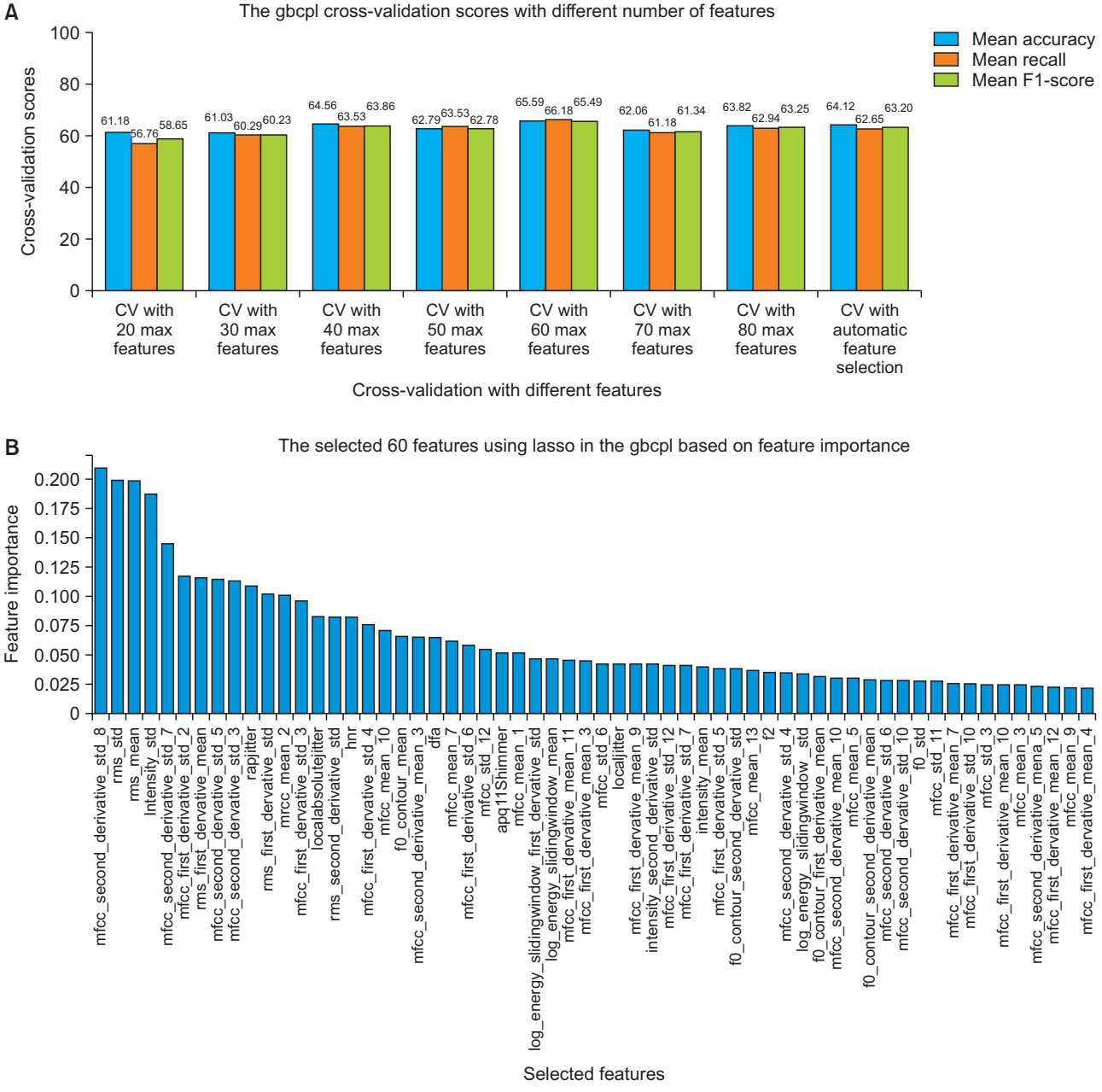

Figure 4. (A) The gbcpl cross-validation performance using a different subset of features and (B) the final 60 chosen features.

through token authentication. Its initial design is developed using HTML, CSS, and Bootstrap and served using WhiteNoise. We used Gunicorn for deployment, a pure-Python WSGI HTTP server. The entire project was then containerized using Docker in an Ubuntu image and deployed to Heroku, a platform-as-a-service cloud provider. Our web application is accessible via the following link: https://pdpredict.herokuapp.com, and its architecture is presented in Figure 3B.

III. Results

Figures 4 and 5 present the performance of the pipelines using nested CV by varying the maximum number of features to select by the meta-transformer in each pipeline based on features importance. We conclude that a maximum of 60 features yielded the best performance in both pipelines.

Table 5 presents the performance of the two pipelines using nested CV and their performance on the training and the holdout sets. From CV performance, we can conclude that gbcpl had higher accuracy, sensitivity, and F1-score than gbcpen, with slight differences of 0.65%, 1.18%, and 0.61%, respectively.

The performance of a model on the training set is not practically useful for model selection, but it is helpful to identify

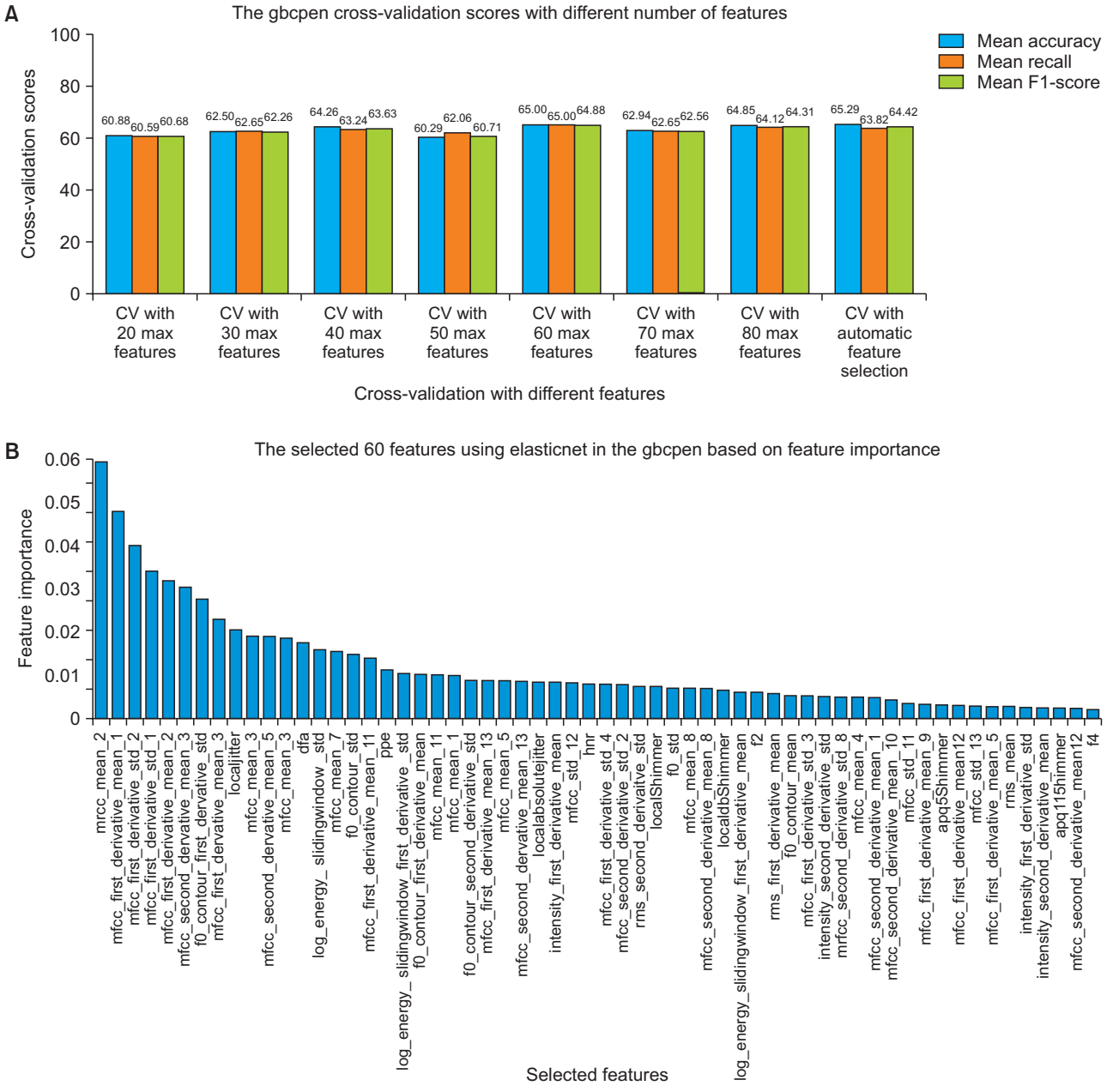

Figure 5. (A) Performance of gbcpen in cross-validation using different subsets of features and (B) the final 60 chosen features.

 https://doi.org/10.4258/hir.2022.28.3.210

Table 5. Summary of the performance of the pipelines

ML pipeline	Accuracy (%)	Recall (%)	F1-score (%)
gbcpl			
Nested cross-validation	65.59 (0.0675)	66.18 (0.1118)	65.49 (0.076)
Training set	71.76 ± 3.38	72.65 ± 3.35	72.01 ± 3.37
Holdout set	71.43 ± 6.83	72.62 ± 6.74	71.76 ± 6.81
gbcpen			
Nested cross-validation	65.00 (0.0587)	65.00 (0.0837)	64.88 (0.0611)
Training set	72.65 ± 3.35	70.29 ± 3.43	71.99 ± 3.38
Holdout set	67.86 ± 7.06	67.86 ± 7.06	67.86 ± 7.06

Nested cross-validation results are presented as mean (standard deviation); training and holdout set performances are reported with 95% confidence intervals.
ML: machine learning.

whether the model has overfitted the training set. First, two final pipelines with a set of hyperparameters were chosen, as shown in Table 4. Those pipelines were trained using the entire training set; then, we measured their performance on the same set to identify overfitting. The following results are reported with 95% confidence intervals (CIs), a range of values calculated from the data, most likely including the actual value estimated for the population. In terms of accuracy, given the sample, gbcpl had an accuracy of 71.76% (95% CI, 68.38%–75.14%). In other words, there is a 95% likelihood that the range of 68.38% to 75.14% covers its accuracy. The accuracy of gbcpen was 72.65% (95% CI, 69.3%–76%). Furthermore, the sensitivity or recall of gbcpl was 72.65% (95% CI, 69.3%–76%). where-as that of gbcpen was lower 70.29% (95% CI, 66.86%–73.72%). The F1-score showed a similar pattern.

Before deciding whether overfitting was an issue, we measured the performance of the pipelines on the holdout data, which was kept unseen during the training process. The performance with unseen data is essential for model selection to draw conclusions regarding the generalizability of a pipeline. Given the sample size, gbcpl was 71.43% (95% CI, 64.6%–78.26%) accurate, 72.62% (95% CI, 65.88%–79.36%) sensitive, and had an F1-score of 71.76% (95% CI, 64.95%–78.57%). Meanwhile, gbcpen was accurate, sensitive and had an F1-score of 67.86% (95% CI, 60.8%–74.92%). From the above, we can conclude that both pipelines generalize well to unseen data and do not overfit the training set.

Both pipelines are saved and deployed to our web server to be accessible via a REST API from a prediction endpoint, with different statuses, a staging status for the gbcpl, and a production status for the gbcpen. The pipelines selection is provided within the desktop application when making predictions.

IV. Discussion

In this study, we presented PD Predict, an ML-based system for predicting and classifying PD. Its architecture is divided into two main components: a client-side desktop application and a server-side web application. PD Predict implements all the steps described in the methodology logically, starting from voice recording to PD classification. One of the advantages of this architecture is its extendibility to other applications than PD prediction. Once the server-side web application development is complete, adding ML pipelines is quickly done in a few lines of code to be available from a prediction endpoint via the REST API. The implemented ML pipelines showed moderate performance, between 65% and 75%, which is to be expected given the quality of the recordings, as the mPower study is a clinical trial performed in uncontrolled environments where most participants recorded their voices at home, outdoors, or in crowded places. In addition, the quality of smartphone microphones is not comparable to that of the sophisticated equipment used in controlled clinical studies and performed in controlled environments. Nevertheless, the pipelines' performance is promising and confirm the usability of smartphone microphones in capturing digital biomarkers of PD.

A client-side application can be anything from a desktop, mobile, to web application. We choose to develop a desktop application for this prototype because both smartphones and laptops' microphones are comparable in performance, which raises the first limitation of this study. For this prototype, the desktop application is not practically useful for patients and healthcare providers, as most of them are not interested in accessing the extracted audio features, which could be easily hidden. However, we decided to keep this application for

illustration purposes. The second limitation is the development of a desktop application instead of a smartphone application because, given the nature of the population, most of the patients are older individuals and may tend to use smartphones more than laptops. The third limitation of this study is the validity of the performance of the ML pipelines in a clinical trial.

Given the above, our subsequent studies will focus on solving the above-acknowledged limitations. First, we aim to develop a cross-platform smartphone application that will be available on both Android and IOS, easy to use, and addressed mainly to patients and healthcare providers. Second, we will also focus on improving the ML pipelines' performance and the server-side web application by adding more useful features. Third, a final study will address scientific evidence and prove the usefulness of the ML pipelines in a clinical trial.

With this prototype of PD Predict, we tried to cross the bridge between research and development, as most ML models are left in papers and never used in practice.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgements

The authors would like to thank every participant who contributed as a user of the Parkinson mPower mobile application and as part of the mPower study [11,12] developed by Sage Bionetworks and described in Synapse (https://doi.org/10.7303/syn4993293).

ORCID

Ilias Tougui (https://orcid.org/0000-0001-7790-4284)
Abdelilah Jilbab (https://orcid.org/0000-0002-1577-9040)
Jamal El Mhamdi (https://orcid.org/0000-0001-8219-3560)

References

1. Weintraub D, Comella CL, Horn S. Parkinson's disease: Part 1: pathophysiology, symptoms, burden, diagnosis, and assessment. Am J Manag Care 2008;14(2 Suppl): S40-8.
2. Goetz CG, Poewe W, Rascol O, Sampaio C. Evidence-based medical review update: pharmacological and surgical treatments of Parkinson's disease: 2001 to 2004. Mov Disord 2005;20(5):523-39. https://doi.org/10.1002/mds.20464
3. Ho AK, Iansek R, Marigliani C, Bradshaw JL, Gates S. Speech impairment in a large sample of patients with Parkinson's disease. Behav Neurol 1998;11(3):131-7. https://doi.org/10.1155/1999/327643
4. Perry B, Herrington W, Goldsack JC, Grandinetti CA, Vasisht KP, Landray MJ, et al. Use of mobile devices to measure outcomes in clinical research, 2010-2016: a systematic literature review. Digit Biomark 2018;2(1):11-30. https://doi.org/10.1159/000486347
5. Sakar CO, Serbes G, Gunduz A, Tunc HC, Nizam H, Sakar BE, et al. A comparative analysis of speech signal processing algorithms for Parkinson's disease classification and the use of the tunable Q-factor wavelet transform. Appl Soft Comput 2019;74:255-63. https://doi.org/10.1016/j.asoc.2018.10.022
6. Gunduz H. Deep learning-based Parkinson's disease classification using vocal feature sets. IEEE Access 2019;7:115540-51. https://doi.org/10.1109/ACCESS.2019.2936564
7. Haq AU, Li JP, Memon MH, Malik A, Ahmad T, Ali A, et al. Feature selection based on L1-norm support vector machine and effective recognition system for Parkinson's disease using voice recordings. IEEE Access 2019;7:37718-34. https://doi.org/10.1109/ACCESS.2019.2906350
8. Almeida JS, Reboucas Filho PP, Carneiro T, Wei W, Damasevicius R, Maskeliunas R, et al. Detecting Parkinson's disease with sustained phonation and speech signals using machine learning techniques. Pattern Recognit Lett 2019;125:55-62. https://doi.org/10.1016/j.patrec.2019.04.005
9. Berus L, Klancnik S, Brezocnik M, Ficko M. Classifying Parkinson's disease based on acoustic measures using artificial neural networks. Sensors (Basel) 2018;19(1):16. https://doi.org/10.3390/s19010016
10. Benba A, Jilbab A, Hammouch A. Analysis of multiple types of voice recordings in cepstral domain using MFCC for discriminating between patients with Parkinson's disease and healthy people. Int J Speech Technol 2016;19(3):449-56. https://doi.org/10.1007/s10772-016-9338-4
11. Bot BM, Suver C, Neto EC, Kellen M, Klein A, Bare C, et al. The mPower study, Parkinson disease mobile data collected using ResearchKit. Sci Data 2016;3:160011. https://doi.org/10.1038/sdata.2016.11
12. mPower: mobile Parkinson disease study [Inter-

net]. Seattle (WA): Sage Bionetworks; 2019 [cited at 2022 Jul 20]. Available from: https://www.synapse.org/#!Synapse:syn4993293/wiki/247859.
13. Synapse REST API: basics table query [Internet]. Seattle (WA): Sage Bionetworks; 2020 [cited at 2022 Jul 20]. Available from: https://docs.synapse.org/rest/org/sagebionetworks/repo/web/controller/TableExamples.html.
14. Wong SL, Gilmour HL, Ramage-Morin PL. Parkinson's disease: prevalence, diagnosis and impact. Health Rep 2014;25(11):10-4.
15. Bonet-Sola D, Alsina-Pages RM. A comparative survey of feature extraction and machine learning methods in diverse acoustic environments. Sensors (Basel) 2021;21(4):1274. https://doi.org/10.3390/s21041274
16. Giannakopoulos T, Pikrakis A. Introduction to audio analysis: a MATLAB approach. Kidlington, UK: Academic Press; 2014.
17. Lenain R, Weston J, Shivkumar A, Fristed E. Surfboard: audio feature extraction for modern machine learning. arXiv [Preprint] 2020 May 18 [Epub]. https://doi.org/10.48550/arXiv.2005.08848.
18. Tougui I, Jilbab A, Mhamdi JE. Impact of the choice of cross-validation techniques on the results of machine learning-based diagnostic applications. Healthc Inform Res 2021;27(3):189-99. https://doi.org/10.4258/hir.2021.27.3.189
19. Kotsiantis SB, Kanellopoulos D, Pintelas PE. Data preprocessing for supervised leaning. Int J Comput Sci 2006;1(2):111-7.
20. Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng 2014;40(1):16-28. https://doi.org/10.1016/j.compeleceng.2013.11.024
21. Zou H, Hastie T. Regularization and variable selection via the elastic net. J R Stat Soc Series B Stat Methodol 2005;67(2):301-20. https://doi.org/10.1111/j.1467-9868.2005.00503.x
22. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13(2):281-305.
23. Vabalas A, Gowen E, Poliakoff E, Casson AJ. Machine learning algorithm validation with a limited sample size. PLoS One 2019;14(11):e0224365. https://doi.org/10.1371/journal.pone.0224365
24. Virbel M, Hansen T, Lobunets O. Kivy: a framework for rapid creation of innovative user interfaces. In: Workshop-Proceedings der Tagung Mensch & Computer 2011; 2011 Sep 11-14; Chemnitz, Germany. p. 69-73.
25. Pham H. PyAudio [Internet]. Cambridge (MA): MIT; 2006 [cited at 2022 Jul 20]. Available from: https://people.csail.mit.edu/hubert/pyaudio/.
26. Cortesi D. PyInstaller Manual [Internet]. [place unknown]: PyInstaller.org; c2020 [cited at 2022 Jul 20]. Available from: https://pyinstaller.org/en/stable/.
27. jrsoftware. Inno Setup [Internet]. [place unknown]: jrsoftware.org; 2022 [cited at 2022 Jul 20]. Available from: https://jrsoftware.org/isinfo.php.
28. Django: the web framework for perfectionists with deadlines [Internet]. [place unknown]: Django Software Foundation; c2022 [cited at 2022 Jul 20]. Available from: https://www.djangoproject.com/.
29. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. J Mach Learn Res 2011;12:2825-30.