Universitat Rovira i Virgili
Escola Tècnica Superior d'Enginyeria

Parkinson's Project

Master in Health Data Science — MHEDAS

Subject: Scientific Programming

Author: Annabel Margalef Burgos, Roger Puig I Arxer, Scarlett Thomas and Ravneet-Rahul Sandhu Singh

Date: January 25, 2025

UNIVERSITAT DE VIC
UNIVERSITAT CENTRAL
DE CATALUNYA

Universitat de Girona

UGA
Université
Grenoble Alpes

Universitat de Barcelona

Universitat de Lleida

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

UNIVERSITAT ROVIRA i VIRGILI

BIOINFORMATICS
BARCELONA

UAB
Universitat Autònoma
de Barcelona

eit Health

Co-funded by the
European Union

Contents

1. Introduction

Parkinson's disease (PD) is a progressive neurodegenerative disorder affecting over 8.5 million people globally as of 2019 [1], with cases rising due to aging populations and improved diagnostics. A significant challenge for about 90% of individuals with PD is speech impairment [2], known as hypokinetic dysarthria, which includes reduced vocal loudness, monotone speech, imprecise articulation, and variable speech rates. Early detection of PD through the analysis of speech can confer immense benefit, including earlier access to traditional and experimental treatment, which can, in turn, help to delay the progression of the disease.

Current methods for detecting PD using speech features include:

- **Support Vector Machines (SVM):** Used for classifying PD patients by analyzing speech features like Mel Frequency Ceptral Coefficients (MFCC) and AutoEncoder patterns [3].
- **Deep Neural Networks (DNN):** Employed to model complex patterns in speech data for accurate PD detection [4].
- **Random Forests (RF):** Utilized to identify important speech features and classify patients based on PD-related symptoms [5].

While these methods demonstrate considerable success, this project aims to explore the effectiveness of K-Nearest Neighbors (KNN) for PD detection. Specifically, three KNN models will be developed: one trained on cleaned data, another on aggregated data, and a third on normalized data. KNN is a simple, non-parametric algorithm that requires minimal assumptions about the data distribution, making it well-suited for small datasets and multi-class problems. Additionally, an API will be created to enable users to test the models by providing their own input values, offering an interactive tool for PD prediction.

Figure 1.1 illustrates project timeline and the task distribution spanning from 8/01/2025 to 19/01/2025.

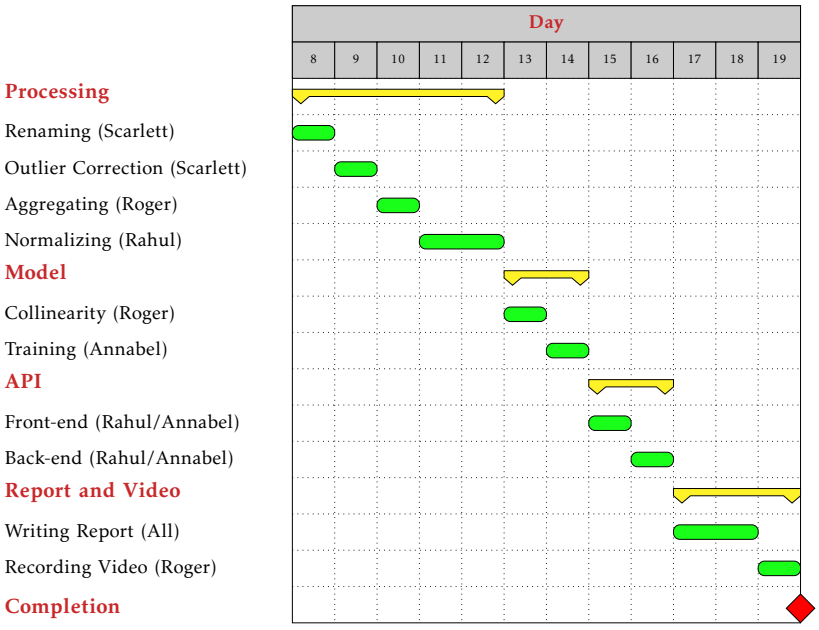

Figure 1.1: Gantt chart of the project.

2. Analysis and Findings

The dataset includes voice measurements from 31 individuals, 23 with Parkinson's disease, across 195 recordings. The recordings are labeled as 0 for healthy and 1 for Parkinson's. Features include: absJitter, apq, apq3, apq5, avFF, dda, dbShimmer, ddp, D2, DFA, HNR, lShimmer, maxFF, minFF, NHR, percJitter, PPE, ppq, rap, RPDE, spread1, and spread2.

First, we conducted an exploratory analysis of the dataset. No missing values, NaN entries, or duplicates were found. We then performed the following preprocessing steps:

- **Renaming:** We designed the function `rename()` to standardize column names. It takes a DataFrame and a dictionary of correct names as inputs, making the dataset easier to use.
- **Outlier Removal:** We used then function `outliers()` based on interquartile range (IQR) method ($Q3 - Q1$) to identify outliers. These were replaced with the mean of non-outlier values within each subject. This avoids mixing healthy and Parkinson's data, assuming values within a subject are consistent.
- **Aggregating:** We used `aggregate()` to group the DataFrame by `subject_id` and compute the mean for each group. This method captures the average feature values across trials for each individual.
- **Normalizing:** Using the function `normalize()`, we applied Min-Max scaling:

$$x_{\text{normalized}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2.1}$$

For the KNN model, we conducted dimensionality reduction, focusing on parameters related to fundamental frequency, jitter, and shimmer. The `collinearity()` function calculated correlations within each feature group and identified the least correlated pairs. To retain a feature, we compared its correlation with all other features. A scoring system assigned higher scores to features with lower overall correlation. Features with higher scores were kept, while others were removed.

Figure 2.1 shows correlations for fundamental frequency features. As you can see, since these metrics are relative to the same measure, it makes sense that they show some sort of correlation between them.

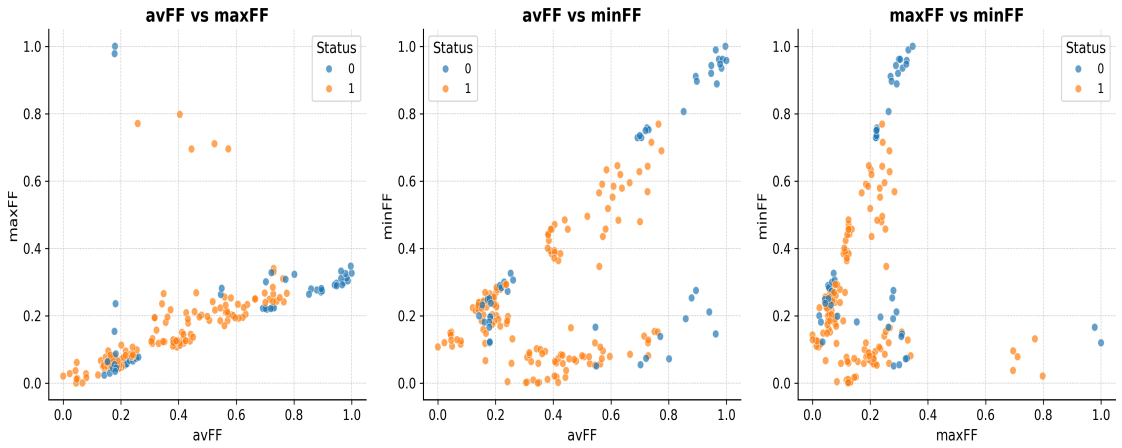

Figure 2.1: Correlation among fundamental frequency-related parameters.

The final selected features to train the model were: absJitter, apq, D2, DFA, HNR, maxFF, NHR, PPE, RPDE, spread1, spread2.

For validation, the data was split into 70% training and 30% testing. To optimize the hyperparameter k, we selected the value where testing and training accuracy were similar. The intersection of these points was found using linear interpolation in the `best_k()` function.

Figure 2.2 compares the training and testing accuracy of the KNN model across three preprocessing methods: raw preprocessed model (`df_clean`), averaged model (`df_avg`), and normalized model (`df_norm`). For the raw preprocessed model, the optimal $k \approx 12$ achieves a balance between testing and training accuracy. In the averaged model, accuracy trends remain flat with no convergence, likely due to the loss of information caused by aggregating data and reducing the initial dataset. In contrast, the normalized model achieves better results, with an optimal $k \approx 4$, balancing accuracy and stability across training and testing datasets.

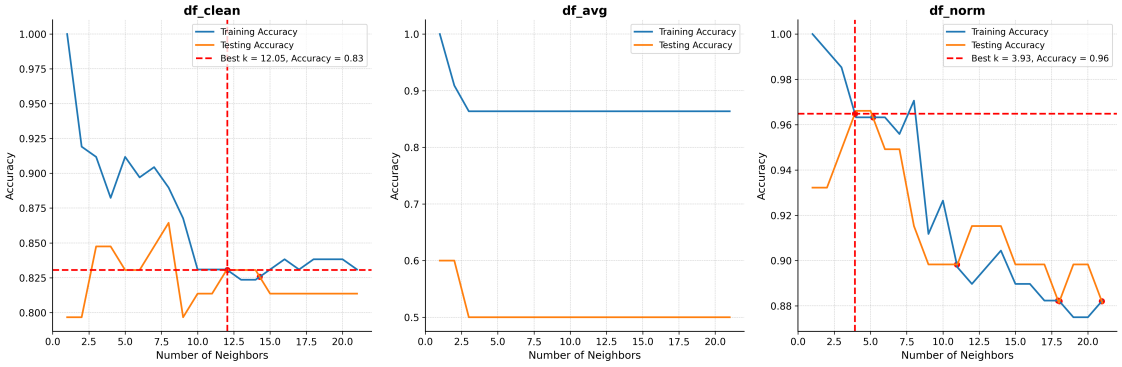

Figure 2.2: Optimal k determination using testing and training accuracy.

Finally, the last step is to integrate the whole process within an API. The backend of this system is implemented using FastAPI. Its primary functions include serving the frontend, managing model files, retrieving evaluation metrics, and handling prediction requests, as shown in **Figure 2.3**. The root endpoint (`/`) serves the `index.html` file, which provides the user interface for interaction. The system organizes models into datetime-labeled folders under `MODEL_DIR`, where each folder represents a distinct training session and contains model files (`.pkl`) and evaluation metrics stored in a `metrics.txt` file.

```
INFO:     Started server process [13222]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:53832 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:53832 - "GET /datetimes/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:53832 - "GET /models/?datetime_folder=2025-01-17_00-43-59 H
TTP/1.1" 200 OK
INFO:     127.0.0.1:53832 - "GET /metrics/?datetime_folder=2025-01-17_00-43-598
model_name=model_avg.pkl HTTP/1.1" 200 OK
INFO:     127.0.0.1:53832 - "GET /metrics/?datetime_folder=2025-01-17_00-43-598
model_name=model_norm.pkl HTTP/1.1" 200 OK
INFO:     127.0.0.1:53832 - "GET /metrics/?datetime_folder=2025-01-17_00-43-598
model_name=model_clean.pkl HTTP/1.1" 200 OK
INFO:     127.0.0.1:55354 - "POST /predict/ HTTP/1.1" 200 OK
```

Figure 2.3: Example of messages exchange between frontend and backend.

The `/datetimes/` endpoint retrieves all datetime-named folders, enabling users to explore different model versions. Once a folder is selected, the `/models/` endpoint lists available model files within it. The `/metrics/` endpoint fetches and displays the content of the `metrics.txt` file for the chosen folder, allowing users to view the performance metrics of the associated models. The prediction process is handled by the `/predict/` endpoint, where the user provides feature inputs and selects a model. The backend then loads the specified model using the `pickle` module and if the selected model is `model_norm` it applies min-max normalization, using pre-computed max-min values from the normalized dataset. Predictions are generated by the model and returned as a JSON response, indicating whether the input data corresponds to a healthy or Parkinson's individual.

The frontend was built using HTML, CSS, and JavaScript. **Figure 2.4** shows a preview of it. Users can select datetime folders and models through dynamically populated dropdown menus, which are updated by fetching data from the backend endpoints `/datetimes/` and `/models/`. Additionally, users can upload a JSON file containing feature values, and the uploaded data automatically populates the input fields, simplifying the data entry process. Metrics for the selected model are retrieved via the `/metrics/` endpoint and displayed directly on the page to provide insights into the model's performance.

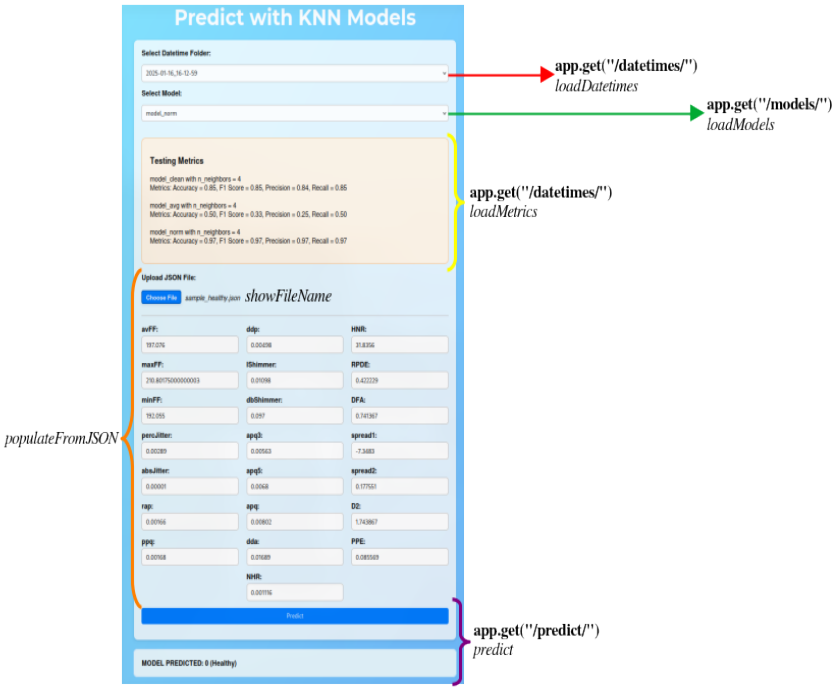

Figure 2.4: Frontend preview with annotated elements: functions related to the backend are shown in **bold**, while functions specific to the frontend are highlighted in *italics*.

The frontend handles form submissions by sending a POST request to the `/predict/` endpoint with the user-provided inputs and the selected model. The backend processes the request, and the prediction results are displayed directly on the webpage. JavaScript functions manage dynamic interactions, such as loading datetime folders, model files, and metrics, as well as validating uploaded JSON files and showing prediction outputs.

References

[1] World Health Organization. Launch of who's parkinson disease technical brief. Accessed: 2025-01-15. URL: https://www.who.int/news/item/14-06-2022-launch-of-who-s-parkinson-disease-technical-brief.

[2] Indiana Parkinson Foundation. Speech and swallowing disorders in parkinson disease, n.d. Accessed: 2025-01-15. URL: https://www.indianaparkinson.org/speech-and-swallowing-disorders-in-parkinson-disease.

[3] Springer. Speech processing for early pd diagnosis, 2023. Accessed: 2025-01-15. URL: https://link.springer.com/article/10.1007/s10772-023-10070-9.

[4] AIP Publishing. Parkinson's disease detection from voice using artificial intelligence, 2024. Accessed: 2025-01-15. URL: https://pubs.aip.org/aip/acp/article/3232/1/040010/3316679/Parkinson-s-disease-detection-from-voice-using.

[5] Springer. Machine learning for pd diagnosis using speech analysis, 2023. Accessed: 2025-01-15. URL: https://link.springer.com/article/10.1007/s13278-022-00905-9.