המחלקה להנדסת תוכנה/המחלקה להנדסת חשמל/המחלקה להנדסה מכנית/המחלקה להנדסת תעשיה וניהול/המחלקה להנדסה רפואית

שם הפרויקט: אבחון סימנים של מחלת פרקינסון באמצעות אות הדיבור

Project Name: Diagnosis of Parkinson's Disease in Human Using Voice Signals

final report-דוח מסכם

שם הסטודנט: יזיד בישארה

מספר תעודת זהות:

שם המנחה: ירמיהו האופטמן

חתימת המנחה:

תאריך ההגשה: ______________________________

Table of Contents

Executive Summary

The goal of this project is to design an easy to and accessible software application that can detect Parkinson disease using speech analysis.

It is estimated that seven to ten million people worldwide currently suffer from PD. Since PD is manifested in speech, it is possible to diagnose Parkinson’s disease from the voice signals

Clinically, there is no specific test that gives defend proof that the PD exist, and therefore PD detection might be difficult and time consuming. Detecting PD from the speech signal can help early detection of the disease , by sending patients with vocal signs to deeper clinical examination, therefor reducing waiting time of the patient.

There has been considerable recent research into the connection between PD and speech impairment. Recently, a wide range of speech signal processing algorithms (dysphonia measures) aiming to predict PD symptom severity using speech signals have been introduced.

Like any classification problem our work will be in three phases first phase is pre-processing in this phase we will cut irrelevant information, second phase is feature extraction we will choose a feature which have correlation with PD speech disorder, the third phase is finding the right classifier to our problem and fitting it to our problem.

From here we will try to find what is the accuracy of our classifier and try to make it more accurate, optimizing feature selection and machine learning classifier algorithm.

For example the program could be used by phone health services, that will ask the patient to read a line or some speech task and try to see could it be Parkinson's or something else.

introduction

Parkinson's disease (PD) is neurodegenerative disease seen in at least 1% individuals 70 years and older. This disease is a direct result of loss of dopamine producing cells throughout the brain. The cells that generate dopamine are located in substantia-nigra part of the brain starts do die of, that way Parkinson's disease and its symptoms start. [1]

It is characterized by a tetrad of symptoms consisting of tremors, rigidity, bradykinesia and postural instability. The disorder in speech and voice are common in early stages almost 60% to 80% of the PD patients report speech disorders. The number increases to 100% in advanced stages of the disease. [2]

If the PD was diagnosed early it will give great advantage in slowing the progress of the disease, the treatment will be more effective. The problem here is that it is hard to diagnose the PD especially in in its early stages, the rate of PD misdiagnosis is approximately 10-25%. [2]

The aim of this project is to study the voice profiles of patients with Parkinson's disease (PD), finding distinctive temporal and spectral features of PD in the voice and using classification algorithms to detect Parkinson's disease (PD) speakers and severity of the disease from healthy controls (HC). This algorithm will be based upon 60 records of PD patients and will be compared to 60 records of healthy people which were recorded by speech therapists in ichilov medical center.

Speech production

Physical model voice production mechanism can be divided into three parts: lungs, vocal folds, and vocal tract. lungs work as source for air flow and pressure. the vocal tract shapes the spectrum of the of the sound as acoustic filter. sound is radiated to the surrounding air at the lips and nostrils. Lungs are equivalent to source of sound & vocal tract equivalent to source filter model for sound production. [3]

Air is pushed from vocal cord to lungs through the vocal tract and out of mouth Working Air is pushed from your lung through your vocal tract and out of your mouth comes speech. [3]

For some voiced sound vocal cords vibrate (open and close). the rate which the vocal cords vibrates determines the pitch of the voice, women and children usually have higher pitch(fast vibration), and adults males have lower pitch(lower vibration). in certain fricatives and plosive (or unvoiced) sound, the vocal cords do not vibrate but remain constantly opened. [3]

Mathematical model

to study the function of speech production mechanism is to categorize speech into three parts. voiced sounds which is excited by the fluctuation of vocal folds/ unvoiced sound where the sound excitation is turbulent noise.

This model is often referred to by LPC model.

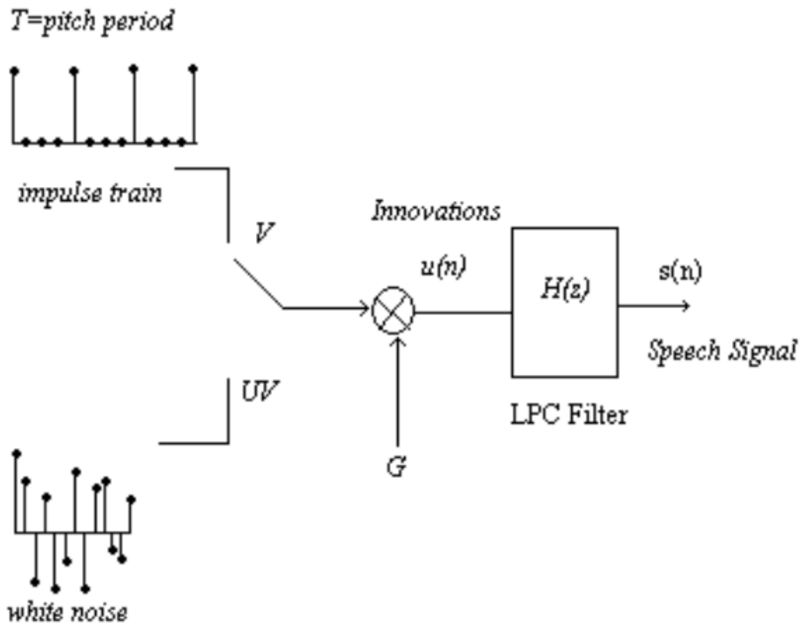


Figure 1:DSP model of speech production [3]

Vocal Cord Vibration $\Longleftrightarrow$ V (voiced)

Fricatives and Plosives $\Longleftrightarrow$ UV (unvoiced)

Vibration Period $\Longleftrightarrow$ T (pitch period)

Air Volume $\Longleftrightarrow$ G (gain)

Air $\Longleftrightarrow$ U(N)(Innovations)

Vocal Tract $\Longleftrightarrow$ H(Z) (LPC Filter)

Figure 2:physical to dsp featuer anlogue[3]

Effect of Parkinson disease on voice characteristic

Speech disorder in Parkinson (also known as part ofhypokinetic Dysarthria),the main cause of the disorder is muscle disorders in the speech mechanism due to damage of the central nervous system.the main problems that can be noticed happens due to paralysis, weakness, or disobedience of speech muscles, The disorder increases with time as the disease progresses and can have tangible effect on communication skills. [4]

The main speech disorders are vocal loudness and vocal decay, meaning that over time there is a fading in the loudness. This Characteristic commonly known as hypophonia, hoarseness, harshness, and especially breathiness in the voice, reduced pitch, loudness inflection, the imprecise consonants and distorted vowels that make speech sound very mumbled and slurred. [4]

Another distinctive characteristic to the disorder is short rushes of speech which is called festinations. An inappropriate pauses and hesitation when beginning to speak, as well as some dysfluencies and voice tremor, especially when the patient is asked to sustain phonation over time.

There are three parts of that effect's speech that relevant to us we will talk about them phonation, articulation, and prosody:

- Phonation is the vibration of the vocal folds to create sound.the usual measurements are performed during sustained vowel phonation and include measurement of F0 (the fundamental frequency or pitch of vocal oscillations), jitter (extent of variation of voice range), shimmer (the extent of variation of expiratory flow), and NHR ratios (the amplitude of noise relative to tonal components in the speech) and voice onset time (VOT). [4]
- Articulation is moving speech organs (e.g., tongue) in the the production of the sound. based on previous studies articulatory deficits may have been partially the result of inadequate tongue elevation and constriction for stops and fricatives. The most common method of evaluating articulatory skills is that of the diadochokinetic (DDK) task. In this test Subjects usually are asked to repeat a combination of the three-syllable item, for example, /pa/-/ta/-/ka/, as fast and long as possible. [4]
  Several patients have demonstrated disorders in the ability to make rapid articulator movements for DDK tasks. Other measurements found differences in vocal tract resonances (i.e., formants), indicating increased variability of the first and second formant (F1 and F2, respectively) frequency.
- Prosody is the change in loudness, pitch, and timing that comes with natural speech. In almost all the cases Prosodic measures are determined from continues speech and include measurement of F0, intensity (relative loudness of speech), articulation rate, pause characteristics, and rhythm. A decreasing pitch range in PD has been noted during the reading task and various changes in speech rate and

pause characteristics have also been found in people with PD. Prosodic intensity changes have also been examined, when PD patients produced significantly smaller intensity variation compared to normal speakers during the reading of a standard passage. Overall, patients with PD demonstrate production defects in all these measurements, including reduced frequency and intensity variations, and differences in speech rate and pause characteristics in reading tasks. [4]

Project goals objectives and indicators

Project goals

The main goal of the project is to build classifier for diagnosis of Parkinson's Disease in Human Using Voice Signals, that can easily be used and trying to detect patients in early stages of the disease.

Project objectives

- The accuracies in literature range between 80% to 99% it depends on the language and speech task, so our main objective is building a classifier with success rate more than 80%.
- Finding extraction methods and classification algorithms that will help in further research.

Indicators

- We will test the classifier on records of healthy controls and PD patients and see if the success rate is above 80%.
- We will see which classifiers and features gives us the best success rate

Assumptions

Great amount of research had been conducted and found correlation between Parkinson's disease and distinctive disorder in voice.

Literature review

dataset

Speech test there is a lot of different speech test in literature,

In one article the data set of 27 individual diagnosed with PD and 27 healthy controlled the average disease stage of1.85 ± 0.55 according to H&Y (hoen and yahr) scale [2]

Other articles used dataset from physionet public database, the data contains 15 PD patients and healthy control group of 16 people. [5]

Our data consists of varios of speakers with ten speech tasks we picked the task for reading text.

More information about the dataset is in the system design section

feature selection

feature selection has been Widley investigated for different purposes, this feature helps us when we want to reduce the dimensionality of the data, by reducing the dimensionality of the data it will be easier and more efficient to build a classifier.

There are two main types of data selection feature extraction and feature selection, feature selection reduces dimensionality by selecting subset of original data, while features extraction transforms the original data to generate other features which are more significant. In our case we will focus more on feature extraction methods.

features that that have been used in the paper:

jitter

Jitter is defined as the parameter of frequency variation from cycle to cycle, and shimmer relates to the amplitude variation of the sound wave. In figure 1 it can be seen the representation of these parameters [19]

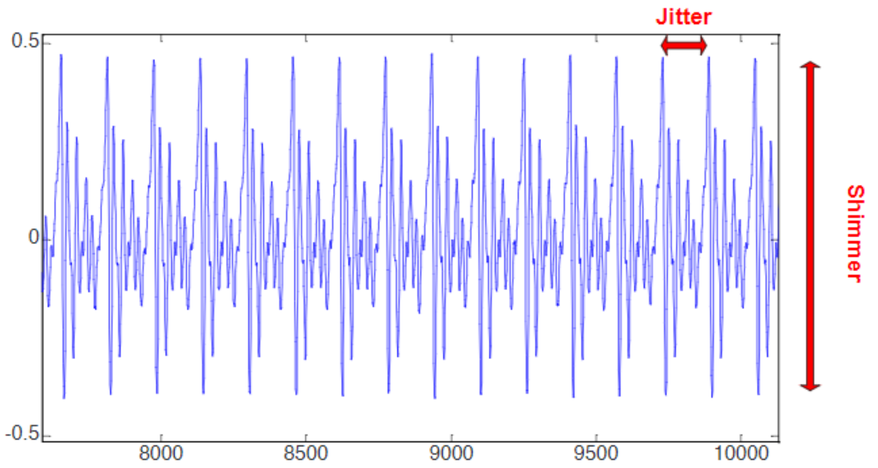


Figure 5:shimmer and jitter axis place [19]

The algorthim removes the linear trends of the signal and then uses a moving average with length
corresponding to about 20 ms (a length similar to one glottal period). Then the peak is searched as the
maximum of the acoustic signal under a window of 15 samples before and 15 samples after the index of the maximum of the moving average.
The HNR is an assessment of the ratio between periodic components and non periodic component comprising a segment of voiced speech, as Murphy and Akande [6]. The first component arises from the

vibration of the vocal cords and the second follows from the glottal noise, expressed in dB. The evaluation between the two components reflects the efficiency of speech, i.e., the greater the flow of air expelled from the lungs into energy of vibration of vocal cords. In these cases the HNR will be greater. A voice sound is thus characterized by a high HNR, which is associated with sonorant and harmonic voice. A low HNR denotes an asthenic voice and dysphonia. [19]

Jitter(absolute): Represents the average absolute difference between two consecutive periods and is known as jitta. The threshold value to detect pathologies in adults is 83.2 μs. [19]

$$jitta = \frac{1}{N-1}\sum_{i=1}^{N-1}|T_i - T_{i-1}|$$

Figure 6: jitter equation

Jitter (local): Represents the average absolute difference between two consecutive periods, divided by the average period. It is known as jitt and has 1.04% as the threshold limit for detecting pathologies.

$$jitt = \frac{jitta}{\frac{1}{N}\sum_{i=1}^{N} T_i} * 100$$

Figure 7: jitter local equation

Where T_i is the duration in seconds of each period and N is the number of periods. [19]

Jitter (rap): Represents the average for the disturbance, i.e., the average absolute difference of one period and the average of the period with its two neighbors, divided by the average period.

$$rap = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}\left|T_i - \left(\frac{1}{3}\sum_{n=i-1}^{i+1} T_n\right)\right|}{\frac{1}{N}\sum_{i=1}^{N} T_i} * 100$$

Figure 8: jitter rap equation

Jitter (ppq5): Represents the ratio of disturbance within five periods, i.e., the average absolute difference between a period and the average containing its four nearest neighbor periods, i.e. two previous and two subsequent periods, divided by average period[19]

$$ppq5 = \frac{\frac{1}{N-1}\sum_{i=2}^{N-2}\left|T_i - \left(\frac{1}{5}\sum_{n=i-2}^{i+2} T_n\right)\right|}{\frac{1}{N}\sum_{i=1}^{N} T_i} * 100$$

Figure 9: jitter ppq5 equation

Jitter (ddpJitter): *Difference of Differences of Periods*, a jitter measure defined as the average absolute difference between the consecutives differences between consecutive intervals, divided by the average interval (an interval is the time between two consecutive points). [19]

$$DDP = \frac{\frac{1}{N-2}\sum_{i=2}^{N-1}|(T_{i+1}-T_i) - (T_i-T_{i+1})|}{\frac{1}{N}\sum_{i=1}^{N} T_i}$$

Figure 10: jitter ddpJitter equation

Shimmer (local): Represents the average absolute difference between the amplitudes of two consecutive periods, divided by the average amplitude. It's called a shim and this parameter was 3.81% as the limit for detecting pathologies. [19]

$$shim = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}|A_i - A_{i+1}|}{\frac{1}{N}\sum_{i=1}^{N} A_i} * 100$$

Figure 11: simmer local equation

Shimmer (local, dB): Represents the average absolute difference of the base 10 logarithm of the difference between two consecutive periods and it is call ShdB. [19]

$$shidB = \frac{1}{N-1}\sum_{i=1}^{N-1}\left|20 * log\left(\frac{A_{i+1}}{A_i}\right)\right|$$

Figure 12: simmer local dB equation

Shimmer (apq3): represents the quotient of amplitude disturbance within three periods, in other words, the average absolute difference between the amplitude of a period and the mean amplitudes of its two neighbors, divided by the average amplitude. [19]

$$apq3 = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}\left|A_i - \left(\frac{1}{3}\sum_{n=i-1}^{i+1} A_n\right)\right|}{\frac{1}{N}\sum_{i=1}^{N} A_i} * 100$$

Figure 13: simmer apq3 equation

Shimmer (apq5): Represents the ratio of perturbation amplitude of five periods, in other words, the average absolute difference between the amplitude of a period and the mean amplitudes of it and its four nearest neighbors, divided by the average amplitude. [19]

$$apq5 = \frac{\frac{1}{N-1}\sum_{i=2}^{N-2}\left|A_i - \left(\frac{1}{3}\sum_{n=i-2}^{i+2} A_n\right)\right|}{\frac{1}{N}\sum_{i=1}^{N} A_i} * 100$$

Figure 14: simmer apq5 equation

HNR: The implementation of the harmonic to noise ratio starts by the detection of the autocorrelation function of the voice signal:

$$AC_{(T)} = \sum_{n=1}^{N} X(n) * X(n+T)$$

Figure 16: autocorrilation equation

The AC(T) is the peak at the index position corresponding to the period of the signal. Therefore the expected values for F0 define a position for that peak between two indices. for the first index (fs/F0max) and the second index (fs/F0min). After determining the indices the local maximum is found within the first and second index, finding their respective amplitude.

$$HNR = 10 * log_{10}\frac{AC(T)}{AC(0) - AC(T)}$$

Figure 17: HNR equation

Despite the usage of the same mathematical formula the algorithms are different also because of the length of used segments or even because of the usage of several segment. [19]

classifier

the goal of classification is building a model that separates data into distinct classes. This model is built by inputting a set of training data for which the classes are pre-labeled in order for the algorithm to learn from. The model is then used by inputting a different dataset for which the classes are withheld, allowing the model to predict their class membership based on what it has learned from the training set. Well-known classification schemes include decision trees and Support Vector Machines. As this type of algorithm requires explicit class labeling, classification is a form of supervised learning.

SVM

Theory

Let's consider that we have L training points, where each input X_i has D attributes(i.e. is of dimensionalty D) and the two classes -1 or +1 (y_i is the label), for example the training set is form of:

$$\{xi,yi\} \quad \text{where } i=1,2\ldots.L \qquad yi \in \{-1, 1\},\ x\in R^2$$

Here we assume the data is linearly separable, meaning that we can draw a line on a graph of x1 vs x2 separating the two classes when D = 2 and a hyperplane on graphs of x1, x2 . . . xD for when D > 2.

This hyperplane can be describes by w*x+b=0 where:

- w is normal hyperplane.
- $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin.

Support Vectors are the examples closest to the separating hyperplane and the aim of Support Vector Machines (SVM) is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes[17]

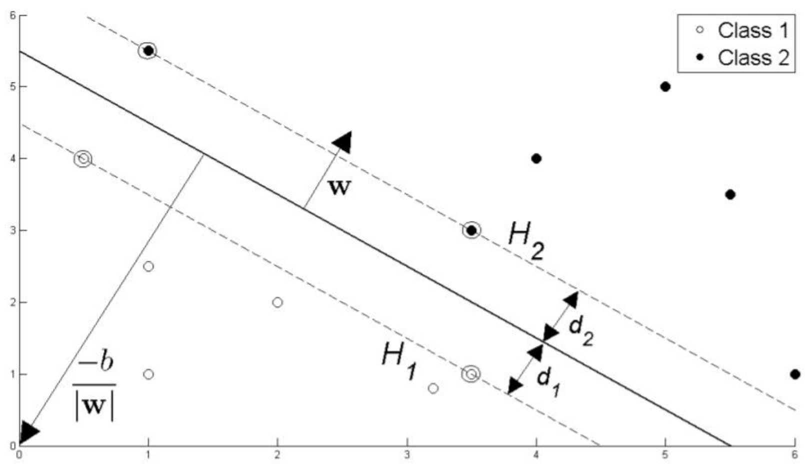


Figure 18:hyperplane through two linear classes [17]

Application

In order to use an SVM to solve a classification or regression problem on data that is not linearly separable, we need to first choose a kernel and relevant parameters which you expect might map the non-linearly separable data into a feature space where it is linearly separable. This is more of skill and knowledge than an exact science and can be achieved empirically - e.g. by trial and error. Know and widely used kernels are polynomial, rbf and linear. [17]

First step consist of choosing a kernel we want to map the linear representation to the new kernel representation $X \rightarrow \emptyset(X)$

for classification, we would need to:

- create H, where $\text{where Hij} = \text{yiyj}\emptyset(Xi) * \emptyset(Xj)$
- Choose how significantly misclassifications should be treated, by selecting a suitable value for the parameter C.
- Find α so that

$$\sum_{i=1}^{L} ai - 0.5a^T H$$

  is maximized, subject to the constraints

$$o < ai \leq C \sum_{i=1}^{L} ai\, yi = 0$$

  This is done using a QP solver.

- calculating$\sum_{i=1}^{L} ai\, yi\emptyset(Xi) = 0$
- Determine the set of Support Vectors S by finding the indices such that

$$o < ai \leq C \sum_{i=1}^{L} ai\, yi = 0$$

For regression, we would then need to:

- Choose how significantly misclassifications should be treated and how large the insensitive loss region should be, by selecting suitable values for the parameters C and ∈.[17]

Performance measure

Performance measure metrics most of the articles measure the performance and validation of the classifier as follows: accuracy, sensitivity, specificity, positive predictive value(PPV),negative predictive value (NPV). [5]

$$\text{accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}},$$

Equation1 :accuarcy equation [2]

$$\text{sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\text{specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

Equation2 :sensitivity and specifity equations [2]

Where TP is true positive, TN is true negative, FP is false positive and FN false negative.

Cross-validation in assessing the general performance [2]

1 Methods

1.1 Study and research

1.1.1 medical

- Research Parkinson disease.
- Research acoustic characteristic on voice.
- Research the effect of Parkinson disease on voice.

1.1.2 Programming and algorithms

- Learn about algorithms and their implementations.
- Learn about signal processing and feature extracting.

1.2 Data analyses and process

- extract distinctive feature in the records.
- building feature extracting model that extract spectral and temporal features.
- comparison between PD patients and healthy controls and based on the features that has been extracted.
- find and design classification algorithm.

1.3 Validation of the results

Comparing results and finding classifier rate of success based on performance measuring metrics, like accuracy, sensitivity and specificity.

1.4 Block diagram

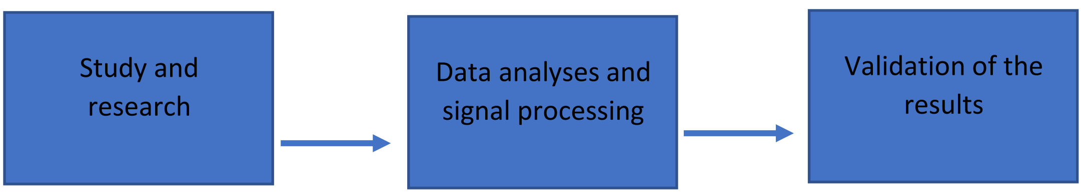
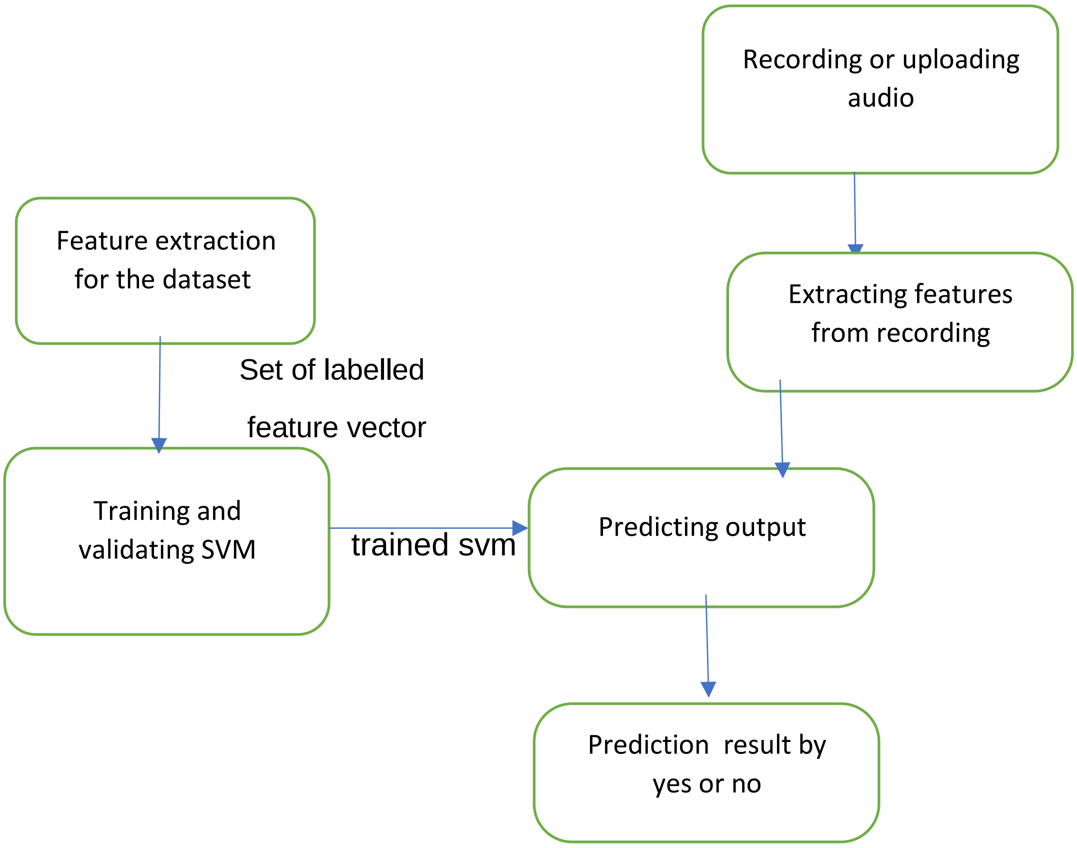


Figure 3:working steps and implemnting[9]

System design and planning

Block diagram



Pre-processing and feature extraction

The dataset that we have consists of several of task recordings we picked the reading test task, the dataset is several recordings of healthy and PD patients people.

our first step is extracting feature from the dataset by building feature extracting function that will go over the wav files and extract the features, in the literature review we talked about the features that we will extract.

The feature we extracted:

The main library we used was parselmouth library in python, parselmouth library aims to provide a complete and pythonic interface to the PRAAT code, PRAAT is a very flexible tool to do speech analysis. It offers a wide range of standard and non-standard procedures, including spectrographic analysis, articulatory synthesis.

all the features were explained in the literature review now we will specify the parameters.

meanF0Hz- Average vocal fundamental frequency
we calculated pitch between 75 to 500 Hz and frequencies above this we ignored, over samples of 20 ms then we took the mean of all the samples.

StdvFh0Hz – standard deviation vocal fundamental frequency
From the pitch that we mentioned above we extracted the standard deviation.

Hnr - harmonic to noise
A Harmonicity object represents the degree of acoustic periodicity, Harmonicity is expressed in dB: if 99% of the energy of the signal is in the periodic part, and 1% is noise, the HNR is 10*log10(99/1) = 20 dB. A HNR of 0 dB means that there is equal energy in the harmonics and in the noise, the Time step we took is 20 ms minimum pitch is 75.

Jitter(%)
Shortest interval is 0.0001 sec(if interval is shorter tan this number it will be ignored), period ceiling 20 ms(50 Hz) longer than this numbe the jiter will be ignored, jitter computation max distance 1.3 sec.

Jitter(Abs): Absolut jitter
The same parameters as Jitter(%)

Rapjitter: Represents the average for the disturbance
The same parameters as Jitter(%)

Ppq5jitter:
The same parameters as Jitter(%)

Localshimmer:
The same parameters as Jitter(%)

Localdbshimmer:
The same parameters as Jitter(%)

Apq3shimmer:
The same parameters as Jitter(%)

Apq5shimmer:
The same parameters as Jitter(%)

ddahimmer: This is the average absolute difference between consecutive differences between the amplitudes of consecutive periods. This is Praat's original **Get shimmer**. The value is three times APQ3.

PPE and DFA: we exracted the spectrogram of the signal then took for PPE took Shannon enthroy, and for DFA by doing **Detrended Fluctuation Analysis from the spectogram**

Describe data:

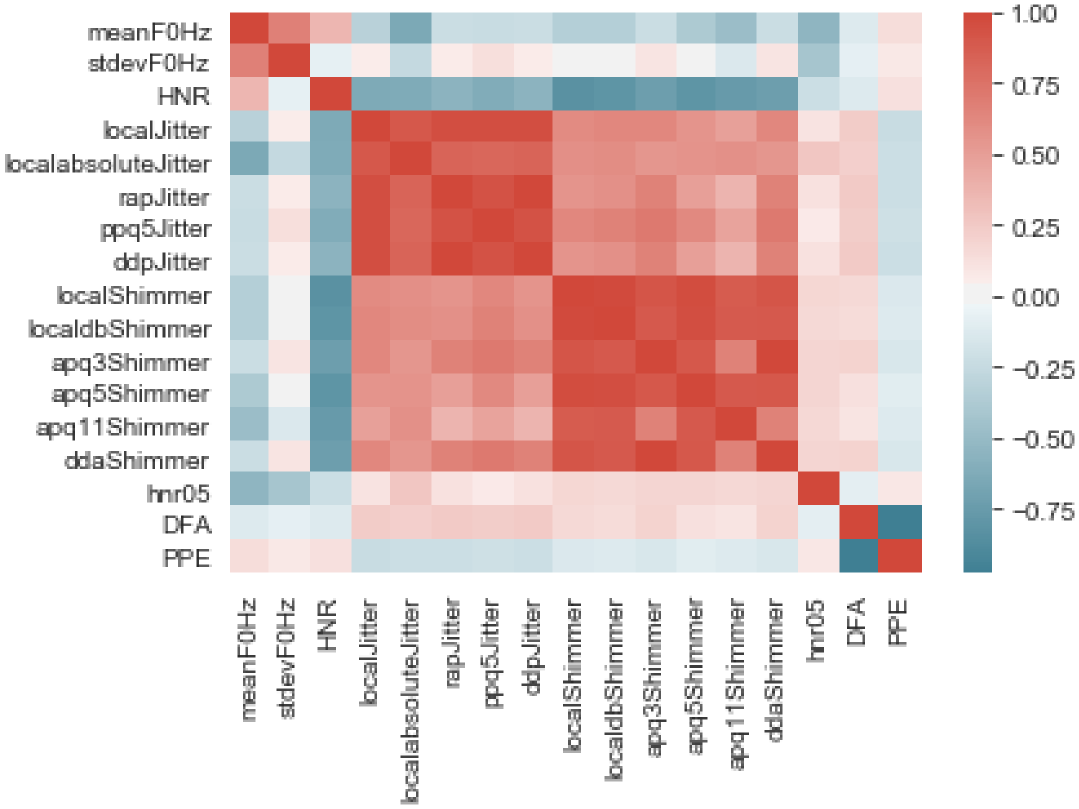
	count	mean	std	min	25%	50%	75%	max
SPKR	134.0	111.634328	67.437584	1.000000	51.250000	118.000000	167.750000	222.000000
PATH_CODE	134.0	0.507463	0.501820	0.000000	0.000000	1.000000	1.000000	1.000000
Gender	134.0	0.395522	0.490797	0.000000	0.000000	0.000000	1.000000	1.000000
Age	134.0	63.708955	10.509448	42.000000	57.000000	64.000000	70.750000	88.000000
meanF0Hz	134.0	149.637916	32.603535	91.940644	123.938054	146.641826	174.330535	236.416853
stdevF0Hz	134.0	25.813600	10.810936	6.308843	17.940982	24.619695	31.961664	77.373378
HNR	134.0	15.970412	3.209565	7.531909	13.501064	16.227916	17.999111	25.424878
localJitter	134.0	0.024111	0.008357	0.012022	0.019241	0.022374	0.026667	0.066146
localabsoluteJitter	134.0	0.000173	0.000086	0.000059	0.000118	0.000152	0.000211	0.000653
rapJitter	134.0	0.010093	0.004842	0.004211	0.007467	0.008908	0.010913	0.039275
ppq5Jitter	134.0	0.010535	0.003767	0.004810	0.008178	0.009632	0.011778	0.026726
ddpJitter	134.0	0.030278	0.014526	0.012632	0.022400	0.026725	0.032740	0.117824
localShimmer	134.0	0.087279	0.020347	0.041546	0.072689	0.086316	0.097245	0.171285
localdbShimmer	134.0	0.830735	0.161594	0.459559	0.726707	0.827771	0.906603	1.500585
apq3Shimmer	134.0	0.030372	0.010803	0.012212	0.023614	0.028052	0.034048	0.075296
apq5Shimmer	134.0	0.042821	0.012750	0.016181	0.034689	0.042079	0.048285	0.108203
apq11Shimmer	134.0	0.087367	0.024610	0.027039	0.071802	0.086602	0.100791	0.200547
ddaShimmer	134.0	0.091116	0.032409	0.036636	0.070842	0.084156	0.102145	0.225888
hnr05	134.0	4.188796	3.091998	-2.365218	2.366554	3.715565	5.299465	17.674606
DFA	134.0	2.012490	0.003070	2.008152	2.010826	2.011396	2.012011	2.027758
PPE	134.0	14.385679	0.982689	11.092096	14.327671	14.712688	14.943130	16.516146

In this dataset there are 68 healthy patients and 66, who recorded text reading.

	Col_value	Count
0	HC	68
1	PD	66

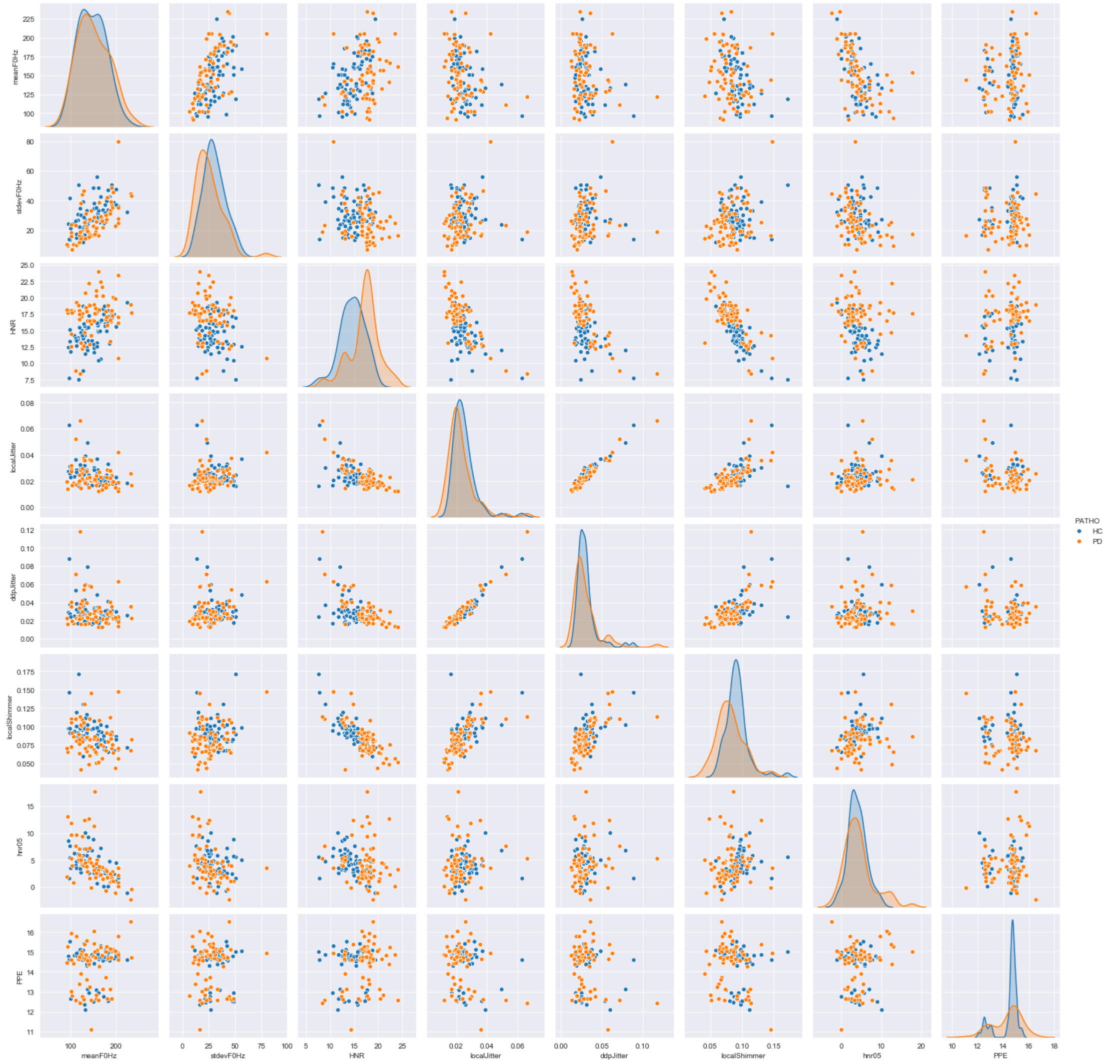
Correlation matrix:

The cross correlation between the features, we can easily see that there is Correlation matrix indicates a similar trend in the measured variables in that jitter and shimmer are closely self correlated and marginally correlated to each other.



Graphical inspection of the variables indicates a similar lack of differentiation in the variables according to outcome, we can see here all the features the blue ones represent the healthy controls and the orange represent PD patients.

Here we took the features that have big difference between them so we can emphasize the difference



We can see that there is high correlation between the jitter features, also between the shimmer features, so they weren’t necessary to use in the end.

Classification

Before starting with training our SVM classifier we will remove features that we couldn't get (NAN), then we will tune the parameters of the SVM classifiers, that we talked about in the literature review.

After this we will scale the features:

we take feature vector and scale by the max and the minimum values in the vector

$$X_{scaled} = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

After scaling we will split the data set into training set and validation set, we split the data 70% train 20% test

After this we started searching for the best parameters for SVM so we tried three kernels: rbf, polynomial, and linear kernel.

we went over all the C parameters from 0.1 to 15 by steps of 0.1 to find the best classifier, so we tried C parameter from 0.1 to 15 by steps of of 0.1, after trying all different parameters and kernels, we found that the best classfier was the palanomial one with degree of 2 and the C parameter is 1.06

```
{'C': 1.06, 'degree': 2, 'gamma': 'scale', 'kernel': 'poly'}
```

The file for finding best parameter

We used 5 cross validation to avoid over fitting we ran every test five times with difrent training and test data set.

```
0.849 (+/-0.101) for {'C': 1.06, 'degree': 2, 'gamma': 'scale', 'kernel': 'poly'}
```

All the the SVC parameters that wwe ran are in log file in final_project\backend\tuning-final.txt

As we can see that the accuracy

```
train score
0.8279569892473119
test score
0.825
              precision    recall  f1-score   support

     healthy       0.88      0.75      0.81        20
        sick       0.78      0.90      0.84        20

    accuracy                           0.82        40
   macro avg       0.83      0.82      0.82        40
weighted avg       0.83      0.82      0.82        40
```

Predicting and extracting feature from record

The function that we used for feature extraction was the same as we used for training the data and we used the classifier that gave the best parameters

The GUI uploading and recording and predicting

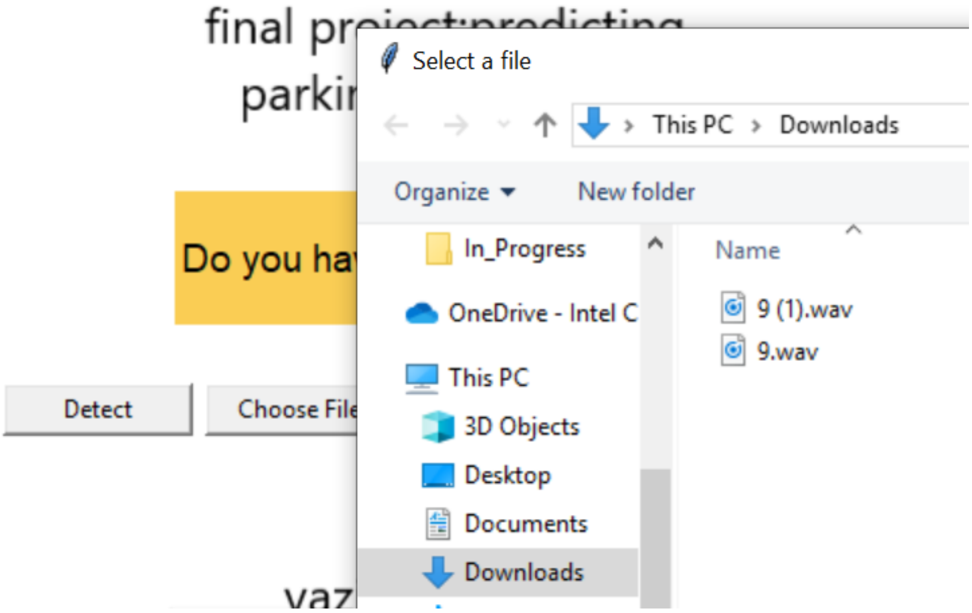
There is a lot of simple gui for recording and uploading records so we used one of them to make it easier for the user

First when we run the software this window pop-up

The user can record his voice by reading the text in Hebrew(רוטב אלף האיים.....).

To start recording the used needs to click on record and to end record the used needs to press on stop recording.

Or the user can upload a recording by clicking on choose a file and pick wav file



And then clickg on detect to predict the output

The output is simple yes or no

2 Project products

Project book

Report with introduction and review of the full algorithms that have been used, literature review, methods that have been used, conclusion and discussions.

Product

Software program that takes as input voice records and calculate probability of the recorded voice to have Parkinson's disease.

Conclusion

Appendix

Feature coronation

In [3]:
```
###########################
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
import seaborn as sns
sns.set_style("darkgrid")
import statsmodels.api as sm
from sklearn import linear_model
#######################################
```

read data

In [4]:
```
path = 'C:\\Users\\yazidbix\\AppData\\Roaming\\final_project\\data_changed.csv'
##read data #########
df = pd.read_csv(path)
df = df[['Dir','FileName','TEXT','SPKR','PATHO','PATH_CODE','Gender','Age','meanF0Hz', 'stdevF0Hz', 'HNR', 'l
ocalJitter', 'localabsoluteJitter', 'rapJitter',
                                            'ppq5Jitter', 'ddpJitter', 'localShimmer', 'localdbShimmer', 'apq
3Shimmer', 'apq5Shimmer',
                                            'apq11Shimmer', 'ddaShimmer','PPE']]
df = df[df.TEXT == '']
```

In [5]:
```
df.describe().transpose()
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
SPKR	279.0	112.820789	66.160070	1.000000	54.000000	119.000000	167.500000	222.000000
PATH_CODE	279.0	0.494624	0.500870	0.000000	0.000000	0.000000	1.000000	1.000000
Gender	279.0	0.383513	0.487115	0.000000	0.000000	0.000000	1.000000	1.000000
Age	279.0	63.637993	10.667715	42.000000	57.000000	64.000000	71.000000	88.000000
meanF0Hz	279.0	151.005914	41.160425	86.701249	119.534364	143.636574	179.117811	256.936094
stdevF0Hz	278.0	10.077768	14.153880	0.870683	3.031489	4.833867	10.199635	120.013224
HNR	279.0	21.798063	5.362055	0.201912	19.211400	22.473739	25.437853	31.751504
localJitter	279.0	0.009329	0.009784	0.001974	0.004133	0.006451	0.009901	0.086575
localabsoluteJitter	279.0	0.000069	0.000085	0.000010	0.000028	0.000044	0.000078	0.000863
rapJitter	279.0	0.004630	0.005926	0.000881	0.001943	0.002910	0.004832	0.056695
ppq5Jitter	278.0	0.004531	0.004529	0.001026	0.002140	0.003128	0.004708	0.035207
ddpJitter	279.0	0.013891	0.017777	0.002643	0.005828	0.008729	0.014497	0.170086
localShimmer	279.0	0.040368	0.033057	0.006872	0.021076	0.029324	0.046817	0.321639
localdbShimmer	279.0	0.395416	0.326055	0.075831	0.197882	0.295945	0.455335	3.370779
apq3Shimmer	278.0	0.018973	0.014678	0.003230	0.009479	0.014039	0.023322	0.099693
apq5Shimmer	278.0	0.021338	0.015716	0.003629	0.011079	0.016381	0.025833	0.095635
apq11Shimmer	277.0	0.029347	0.020740	0.004519	0.016179	0.023139	0.034097	0.124123
ddaShimmer	278.0	0.056920	0.044033	0.009689	0.028437	0.042118	0.069967	0.299078
PPE	279.0	9.156015	0.623027	7.312883	8.761551	9.192293	9.570767	10.939579

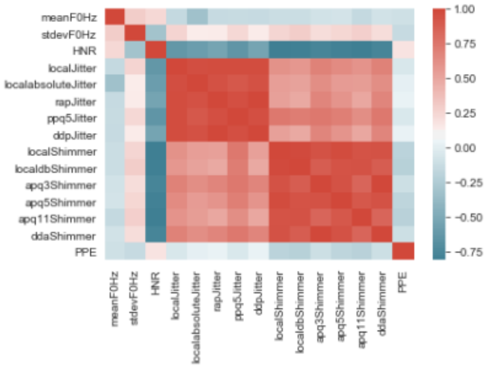
In [6]:
```
df3_count = df.groupby('PATHO').size().reset_index(name='Count').rename(columns={'PATHO':'Col_value'})
print (df3_count)
```

```
  Col_value  Count
0        HC    138
1        PD    141
```

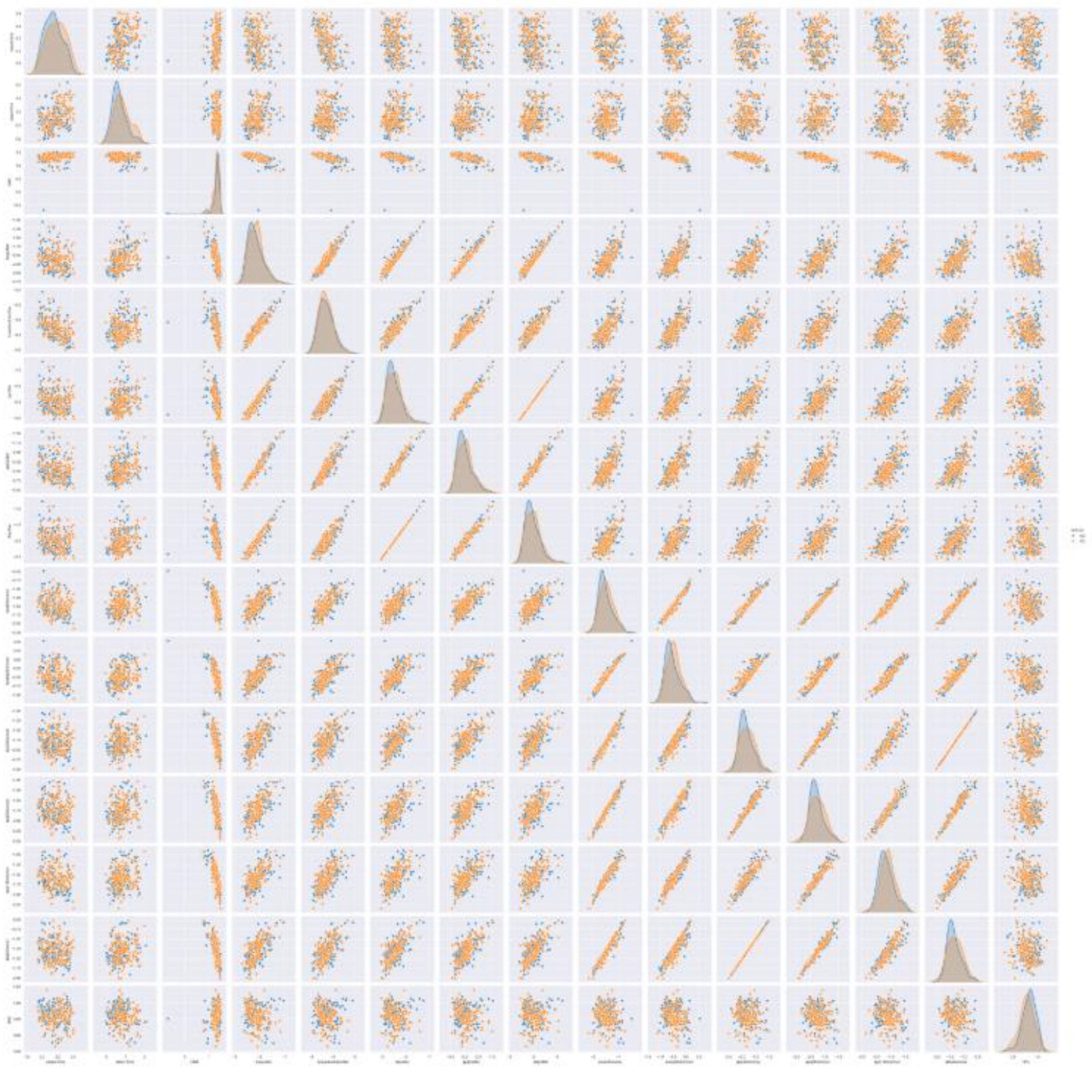
In [8]:
```
features=['meanF0Hz', 'stdevF0Hz', 'HNR', 'localJitter', 'localabsoluteJitter', 'rapJitter',
                                          'ppq5Jitter', 'ddpJitter', 'localShimmer', 'localdbShimmer', 'apq
3Shimmer', 'apq5Shimmer',
                                          'apq11Shimmer', 'ddaShimmer','PPE']

column_list1 = df[features].columns.values.tolist()
cmap = sns.diverging_palette(220,15, as_cmap=True)
correlations1 = df[column_list1].corr()
print ("Correlation matrix")
print (sns.heatmap(correlations1, cmap=cmap,annot_kws={'size':100}))
```

```
Correlation matrix
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [10]:
```
df3_train=df[features]
df3_train_log = df3_train.apply(np.log10)
df3_train_log['PATHO'] = df['PATHO']
df3_train_log.head()
sns.pairplot(df3_train_log, hue= 'PATHO')
sns.plt.show()
```



feature extraction python

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 18 15:09:27 2020

@author: Yazid
"""

import numpy as np
import parselmouth
from parselmouth.praat import call, run_file
import pandas as pd
import nolds
from scipy import signal
from scipy.io import wavfile
from pyentrp import entropy
import sys
```

```
dname='C:\\Users\\Yazid\\Desktop\\final project\\dataset\\'
csvname='PARKINSON2019_BALANCE.csv'
#######################################################
def measurePitch(voiceID, f0min, f0max, unit):
    snd = parselmouth.Sound(voiceID) # read the sound
###############pitch of voice para: frame size(ms) low_freq
high_freq############
    pitch = call(snd, "To Pitch", 0.02, f0min, f0max) #create a
praat pitch object
    ############mean of pitch##############
    meanF0 = call(pitch, "Get mean", 0, 0, unit) # get mean pitch
    ############ std of Pitch ####################
    stdevF0 = call(pitch, "Get standard deviation", 0 ,0, unit) #
get standard deviation
    ####################noise to tonacl coponet frame(ms)
min_freq silence threshhold number of period_ pr_window
#############
    harmonicity = call(snd, "To Harmonicity (cc)", 0.02, 75, 0.1,
1.0)
    hnr = call(harmonicity, "Get mean", 0, 0)
    ################grt pitch  for other
calculations##################
    pointProcess = call(snd, "To PointProcess (periodic, cc)",
f0min, f0max)
    ##############jitter parameters (self, command(ster),
starts(s),end(s),min jitter,maxjitter,frame size(sec),ma distance
between jitters)##############
    localJitter = call(pointProcess, "Get jitter (local)", 0, 0,
0.0001, 0.02, 1.3)
    ###############abs jitter same paramters as
above##############
    localabsoluteJitter = call(pointProcess, "Get jitter (local,
absolute)", 0, 0, 0.001, 0.02, 1.3)
    ###########ssame as above ############################
    rapJitter = call(pointProcess, "Get jitter (rap)", 0, 0,
0.0001, 0.02, 1.3)
    ##################same parmater as
above#####################
    ppq5Jitter = call(pointProcess, "Get jitter (ppq5)", 0, 0,
0.0001, 0.02, 1.3)
    ##################################3same as
above#########################
    ddpJitter = call(pointProcess, "Get jitter (ddp)", 0, 0,
0.0001, 0.02, 1.3)
    localShimmer =  call([snd, pointProcess], "Get shimmer
(local)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    localdbShimmer = call([snd, pointProcess], "Get shimmer
(local_dB)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    apq3Shimmer = call([snd, pointProcess], "Get shimmer (apq3)",
0, 0, 0.0001, 0.02, 1.3, 1.6)
    apq5Shimmer = call([snd, pointProcess], "Get shimmer (apq5)",
0, 0, 0.0001, 0.02, 1.3, 1.6)
    apq11Shimmer =  call([snd, pointProcess], "Get shimmer
(apq11)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    ddaShimmer = call([snd, pointProcess], "Get shimmer (dda)",
0, 0, 0.0001, 0.02, 1.3, 1.6)
    #harmonicity05 = call(snd, "To Harmonicity (cc)", 0.01, 500,
0.1, 1.0)
```

```
    #hnr05 = call(harmonicity05, "Get mean", 0, 0)
    #harmonicity15 = call(snd, "To Harmonicity (cc)", 0.01, 1500,
0.1, 1.0)
    #hnr15 = call(harmonicity15, "Get mean", 0, 0)
    #harmonicity25 = call(snd, "To Harmonicity (cc)", 0.01, 2500,
0.1, 1.0)
    #hnr25 = call(harmonicity25, "Get mean", 0, 0)
    #harmonicity35 = call(snd, "To Harmonicity (cc)", 0.01, 3500,
0.1, 1.0)
    #hnr35 = call(harmonicity35, "Get mean", 0, 0)
    #harmonicity38 = call(snd, "To Harmonicity (cc)", 0.01, 3800,
0.1, 1.0)
    #hnr38 = call(harmonicity38, "Get mean", 0, 0)
    return meanF0, stdevF0,
hnr,localJitter,localabsoluteJitter,rapJitter,ppq5Jitter,ddpJitte
r,localShimmer,localdbShimmer,apq3Shimmer,apq5Shimmer,apq11Shimme
r,ddaShimmer

###################################################
df = pd.read_csv(dname+csvname)## reasinf data
df.drop(df[df['SPKR'] ==217].index,inplace=True)## delteting
missing data
#############################################
Dir=[]
FileName=[]
TEXT=[]
SPKR=[]
PATHO=[]
PATH_CODE=[]
Gender=[]
Age=[]
mean_F0_list = []
sd_F0_list = []
hnr_list = []
localJitter_list = []
localabsoluteJitter_list = []
rapJitter_list = []
ppq5Jitter_list = []
ddpJitter_list = []
localShimmer_list = []
localdbShimmer_list = []
apq3Shimmer_list = []
apq5Shimmer_list = []
apq11Shimmer_list = []
ddaShimmer_list = []
PPE_list = []
for i, row in df.iterrows():
    print(i/14.02,'% done')
    dir_name=dname+'NewRecording'+ row['Dir'][40:]
    file_name='\\'+row['FileName']
    wave_file=dir_name+file_name
    print('getting features
SPKR:',row['SPKR'],'TEXT:',row['TEXT'])
    ##############read recording#############
    sample_rate, samples = wavfile.read(wave_file)
    #######spectogram for PPE feature
    frequencies, times, spectogram = signal.spectrogram(samples,
sample_rate)
```

```
    ###########read recording in prat format for parselmouth
features
    snd = parselmouth.Sound(wave_file)
    #########shannon enthropy in freq domain (PPE feature)
    PPE = entropy.shannon_entropy(times)
    ##########feature extraction praat######
    (meanF0Hz, stdevF0Hz,
HNR,localJitter,localabsoluteJitter,rapJitter

,ppq5Jitter,ddpJitter,localShimmer,localdbShimmer,apq3Shimmer,apq
5Shimmer
     ,apq11Shimmer,ddaShimmer)= measurePitch(snd, 75, 600,
"Hertz")
    print('appending features')
    #########appending all data############
    Dir.append(dir_name) # directory name
    FileName.append(row['FileName'])
    TEXT.append(row['TEXT'])
    SPKR.append(row['SPKR'])
    PATHO.append(row['PATHO'])
    PATH_CODE.append(row['PATH_CODE'])
    Gender.append(row['Gender'])
    Age.append(row['Age'])
    mean_F0_list.append(meanF0Hz) # make a mean F0 list
    sd_F0_list.append(stdevF0Hz) # make a sd F0 list
    hnr_list.append(HNR)
    localJitter_list.append(localJitter)
    localabsoluteJitter_list.append(localabsoluteJitter)
    rapJitter_list.append(rapJitter)
    ppq5Jitter_list.append(ppq5Jitter)
    ddpJitter_list.append(ddpJitter)
    localShimmer_list.append(localShimmer)
    localdbShimmer_list.append(localdbShimmer)
    apq3Shimmer_list.append(apq3Shimmer)
    apq5Shimmer_list.append(apq5Shimmer)
    apq11Shimmer_list.append(apq11Shimmer)
    ddaShimmer_list.append(ddaShimmer)
    PPE_list.append(PPE)
print('creating dataframe')
dtf = pd.DataFrame(np.column_stack([Dir,
FileName,TEXT,SPKR,PATHO,PATH_CODE,Gender,Age, mean_F0_list,
                                    sd_F0_list,
hnr_list,localJitter_list,localabsoluteJitter_list,

rapJitter_list,ppq5Jitter_list,ddpJitter_list,localShimmer_list

localdbShimmer_list,apq3Shimmer_list,apq5Shimmer_list,apq11Shimme
r_list,
                                    ddaShimmer_list,PPE_list]),

columns=['Dir','FileName','TEXT','SPKR','PATHO','PATH_CODE','Gend
er','Age',
                                        'meanF0Hz', 'stdevF0Hz',
'HNR','localJitter','localabsoluteJitter',

'rapJitter','ppq5Jitter','ddpJitter','localShimmer','apq3Shimmer'
,
```

```
'apq5Shimmer','apq11Shimmer','ddaShimmer','PPE'])  #add these
lists to pandas in the right order
##########save data################
dtf.to_csv('C:\\Users\\Yazid\\Desktop\\final
project\\dataset\\all_data.csv')
```

SVM training and tuning

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jul 19 14:44:58 2020

@author: yazidbix
"""

import pandas
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.svm import SVC
import numpy as np
import sys
sys.stdout =
open("C:\\Users\\yazidbix\\AppData\\Roaming\\final_project\\backe
nd\\val_10.txt", "w")

# load the dataset (local path)
path =
'C:\\Users\\yazidbix\\AppData\\Roaming\\final_project\\data.csv'
####### features########
features_all =
['Dir','FileName','TEXT','SPKR','PATHO','PATH_CODE','Gender','Age
','meanF0Hz', 'stdevF0Hz', 'HNR',
                                            'localJitter',
'localabsoluteJitter', 'rapJitter',
                                            'ppq5Jitter',
'ddpJitter', 'localShimmer', 'localdbShimmer', 'apq3Shimmer',
'apq5Shimmer',
                                            'apq11Shimmer',
'ddaShimmer','hnr05','hnr15','hnr25','hnr35','hnr38','DFA','PPE']
###########remove all unwanted features############
features_wanted=['PATH_CODE','meanF0Hz', 'stdevF0Hz', 'HNR',
'localJitter', 'localabsoluteJitter', 'rapJitter',
                                            'ppq5Jitter',
'ddpJitter', 'localdbShimmer', 'apq3Shimmer', 'apq5Shimmer',
                                            'apq11Shimmer',
'ddaShimmer','hnr05','hnr15','PPE']


#####erad data ###############
dataset = pandas.read_csv(path)
#############remove problmatic recording###########
dataset = dataset[dataset.SPKR != 30]
############## keep readinf text task recording##############
dataset = dataset[dataset.TEXT == 'T_ILND']
##########keep wanted features ##############
dat=dataset[features_wanted]
############delete NAN rows############
dat=dat.dropna()
```

```
##### convert from pandas to array ###############
array = dat.values
################ scale by min and max ##############
scaler = MinMaxScaler(feature_range=(0,1))
scaled = scaler.fit_transform(array)
# X stores feature values
X = scaled[:,1:]
# Y stores labels sick/healty
Y = scaled[:,0]

print(__doc__)

# split dataset into training set (75%) and validation set (25%)
X_train, X_test, y_train, y_test = train_test_split(
    X,Y , test_size=0.30, random_state=10,stratify=Y)
rng_c=np.arange(start=0.01,stop=10,step=0.1)
rng_gam=['auto','scale']
rng_deg=np.arange(start=2,stop=5,step=1)

# Set the parameters by cross-validation
tuned_parameters =
[{'kernel':['linear'],'C':rng_c,'gamma':rng_gam},

{'kernel':['poly'],'C':rng_c,'gamma':rng_gam,'degree':rng_deg},
                    {'kernel':['rbf'],'C':rng_c,'gamma':rng_gam}]

scores = ['precision', 'recall']

for score in scores:
    print("# Tuning hyper-parameters for %s" % score)
    print()

    clf = GridSearchCV(
        SVC(), tuned_parameters, scoring='%s_macro' % score
    )
    clf.fit(X_train, y_train)

    print("Best parameters set found on development set:")
    print()
    print(clf.best_params_)
    print()
    print("Grid scores on development set:")
    print()
    means = clf.cv_results_['mean_test_score']
    stds = clf.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds,
clf.cv_results_['params']):
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params))
    print()

    print("Detailed classification report:")
    print()
    print("The model is trained on the full development set.")
    print("The scores are computed on the full evaluation set.")
    print()
    y_true, y_pred = y_test, clf.predict(X_test)
```

```
    print(classification_report(y_true,
y_pred,labels=[0.0,1.0],target_names=['healthy', 'sick']))
    print()
```

predicting and extracting features

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 13 13:55:36 2020

@author: yazidbix
"""


import joblib
import parselmouth
from parselmouth.praat import call, run_file
import pandas as pd
from scipy import signal
from scipy.io import wavfile
from pyentrp import entropy
import numpy as np
import sklearn
import nolds

def loadModel(PATH):
    clf = joblib.load(PATH)
    return clf

def measurePitch(voiceID, f0min, f0max, unit):
    sound = parselmouth.Sound(voiceID) # read the sound
    pitch = call(sound, "To Pitch", 0.0, f0min, f0max) #create a 
praat pitch object
    meanF0 = call(pitch, "Get mean", 0, 0, unit) # get mean pitch
    stdevF0 = call(pitch, "Get standard deviation", 0 ,0, unit) # 
get standard deviation
    harmonicity = call(sound, "To Harmonicity (cc)", 0.01, 75, 
0.1, 1.0)
    hnr = call(harmonicity, "Get mean", 0, 0)
    pointProcess = call(sound, "To PointProcess (periodic, cc)", 
f0min, f0max)#create a praat pitch object
    localJitter = call(pointProcess, "Get jitter (local)", 0, 0, 
0.0001, 0.02, 1.3)
    localabsoluteJitter = call(pointProcess, "Get jitter (local, 
absolute)", 0, 0, 0.0001, 0.02, 1.3)
    rapJitter = call(pointProcess, "Get jitter (rap)", 0, 0, 
0.0001, 0.02, 1.3)
    ppq5Jitter = call(pointProcess, "Get jitter (ppq5)", 0, 0, 
0.0001, 0.02, 1.3)
    ddpJitter = call(pointProcess, "Get jitter (ddp)", 0, 0, 
0.0001, 0.02, 1.3)
    localShimmer =  call([sound, pointProcess], "Get shimmer 
(local)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    localdbShimmer = call([sound, pointProcess], "Get shimmer 
(local_dB)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    apq3Shimmer = call([sound, pointProcess], "Get shimmer 
(apq3)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
```

```
    aqpq5Shimmer = call([sound, pointProcess], "Get shimmer
(apq5)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    apq11Shimmer =  call([sound, pointProcess], "Get shimmer
(apq11)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    ddaShimmer = call([sound, pointProcess], "Get shimmer (dda)",
0, 0, 0.0001, 0.02, 1.3, 1.6)
    harmonicity05 = call(sound, "To Harmonicity (cc)", 0.01, 500,
0.1, 1.0)
    hnr05 = call(harmonicity05, "Get mean", 0, 0)
    harmonicity15 = call(sound, "To Harmonicity (cc)", 0.01,
1500, 0.1, 1.0)
    hnr15 = call(harmonicity15, "Get mean", 0, 0)
    harmonicity25 = call(sound, "To Harmonicity (cc)", 0.01,
2500, 0.1, 1.0)
    hnr25 = call(harmonicity25, "Get mean", 0, 0)
    harmonicity35 = call(sound, "To Harmonicity (cc)", 0.01,
3500, 0.1, 1.0)
    hnr35 = call(harmonicity35, "Get mean", 0, 0)
    harmonicity38 = call(sound, "To Harmonicity (cc)", 0.01,
3800, 0.1, 1.0)
    hnr38 = call(harmonicity38, "Get mean", 0, 0)
    return meanF0, stdevF0, hnr,localJitter, localabsoluteJitter,
rapJitter, ppq5Jitter,ddpJitter, localShimmer, localdbShimmer,
apq3Shimmer, aqpq5Shimmer, apq11Shimmer,ddaShimmer, hnr05, hnr15
,hnr25 ,hnr35 ,hnr38


def predict(clf, wavPath):
    file_list = []
    meanF0_list=[]
    stdevF0_list=[]
    hnr_list=[]
    localJitter_list = []
    localabsoluteJitter_list = []
    rapJitter_list = []
    ppq5Jitter_list = []
    ddpJitter_list = []
    localShimmer_list = []
    localdbShimmer_list = []
    apq3Shimmer_list = []
    aqpq5Shimmer_list = []
    apq11Shimmer_list = []
    ddaShimmer_list=[]
    hnr05_list = []
    hnr15_list = []
    hnr25_list = []
    hnr35_list = []
    hnr38_list = []
    DFA_list = []
    PPE_list = []

    sample_rate, samples = wavfile.read(wavPath)
    frequencies, times, spectogram = signal.spectrogram(samples,
sample_rate)
    snd = parselmouth.Sound(wavPath)
    DFA = nolds.dfa(times)
    PPE = entropy.shannon_entropy(times)
```

```
    (meanF0, stdevF0, hnr,localJitter, localabsoluteJitter,
rapJitter, ppq5Jitter,ddpJitter, localShimmer, localdbShimmer,
apq3Shimmer, aqpq5Shimmer,
     apq11Shimmer,ddaShimmer, hnr05, hnr15, hnr25, hnr35, hnr38)
= measurePitch(snd, 75, 1000, "Hertz")
    meanF0_list.append(meanF0)
    stdevF0_list.append(stdevF0)
    hnr_list.append(hnr)
    localJitter_list.append(localJitter)  # make a mean F0 list
    localabsoluteJitter_list.append(localabsoluteJitter)  # make
a sd F0 list
    rapJitter_list.append(rapJitter)
    ppq5Jitter_list.append(ppq5Jitter)
    ddpJitter_list.append(ddpJitter)
    localShimmer_list.append(localShimmer)
    localdbShimmer_list.append(localdbShimmer)
    apq3Shimmer_list.append(apq3Shimmer)
    aqpq5Shimmer_list.append(aqpq5Shimmer)
    apq11Shimmer_list.append(apq11Shimmer)
    ddaShimmer_list.append(ddaShimmer)
    hnr05_list.append(hnr05)
    hnr15_list.append(hnr15)
    hnr25_list.append(hnr25)
    hnr35_list.append(hnr35)
    hnr38_list.append(hnr38)
    DFA_list.append(DFA)
    PPE_list.append(PPE)
    ##we excluded hnr values  because of the NAN values
    toPred = pd.DataFrame(np.column_stack(
        [meanF0, stdevF0, hnr,localJitter_list,
localabsoluteJitter_list, rapJitter_list,
ppq5Jitter_list,ddpJitter_list, localShimmer_list,
         localdbShimmer_list, apq3Shimmer_list,
aqpq5Shimmer_list,
apq11Shimmer_list,ddaShimmer_list,DFA_list,PPE_list]),
                             columns=['meanF0Hz', 'stdevF0Hz',
'HNR', 'localJitter', 'localabsoluteJitter', 'rapJitter',
                                            'ppq5Jitter',
'ddpJitter', 'localShimmer', 'localdbShimmer', 'apq3Shimmer',
'apq5Shimmer',
                                            'apq11Shimmer',
'ddaShimmer','DFA','PPE'] )  # add these lists to pandas in the
right order
    toPred = toPred.apply(np.log10)

    resp = clf.predict(toPred)
    resp = str(resp)

    if resp == "[1.]":
        return True
    else:
        return False
```

main (GUI)

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 13 13:31:22 2020

@author: yazidbix
"""


import threading
import pyaudio
import wave
import os
from tkinter import *
from tkinter import filedialog
from lib.RecognitionLib import *

global filePath
global clf

path =
r"C:\Users\yazidbix\AppData\Roaming\final_project\lib\trainedMode
l.sav"
clf = loadModel(path)
filePath = "unknow"


# Audio record

class App():
    chunk = 1024
    sample_format = pyaudio.paInt16
    channels = 1
    fs = 44100

    frames = []

    def __init__(self, master):

        self.isrecording = False
        self.button1 = Button(app, text='Recording', width=12,
command=self.startrecording)
        self.button2 = Button(app, text='Stop Recording',
width=12, command=self.stoprecording)
        self.button1.place(x=300, y=200)
        self.button2.place(x=400, y=200)

    def startrecording(self):

        textTest = "Please read this text for the record  רוטב
אלף האיים"


        self.textParkiTest = Label(app, text=textTest,
font=('bold', '13'), bg='red')
        self.textParkiTest.place(x=0, y=270)

        self.textParkiTest4.place(x=0, y=360)
```

```
        self.p = pyaudio.PyAudio()
        self.stream = self.p.open(format=self.sample_format,
channels=self.channels, rate=self.fs,
                                  frames_per_buffer=self.chunk,
input=True)
        self.isrecording = True

        print('Recording')
        global t
        t = threading.Thread(target=self.record)
        t.start()

    def stoprecording(self):
        if self.isrecording == False:
            print("Press Recoring button first")
        else:
            self.textParkiTest.destroy()
            self.textParkiTest1.destroy()
            self.textParkiTest2.destroy()
            self.textParkiTest3.destroy()
            self.textParkiTest4.destroy()

            self.isrecording = False
            print('recording complete')
            self.filename = "recordingAudio.wav"
            wf = wave.open(self.filename, 'wb')
            wf.setnchannels(self.channels)

wf.setsampwidth(self.p.get_sample_size(self.sample_format))
            wf.setframerate(self.fs)
            wf.writeframes(b''.join(self.frames))
            wf.close()

    def record(self):
        self.frames.clear()
        while self.isrecording:
            data = self.stream.read(self.chunk)
            self.frames.append(data)


def chooseFile():
    global filePath
    filePath = filedialog.askopenfilename(initialdir="/C",
title="Select a file", filetypes=[("wav file", "*.wav")])
    print("path :", filePath)


def execAI():
    global part_label1
    global part_label2
    global part_label3
    global filePath
    errmsg = "Path not valid"

    if ((filePath == "unknow") or (filePath == "")) and
os.path.exists("recordingAudio.wav"):
        filePath = "recordingAudio.wav"
```

```
    if (filePath != "unknow") and (filePath != ""):
        if predict(clf, filePath):
            # Test label1
            try:
                part_label1
            except NameError:
                part_label1 = None

            if part_label1 is not None:
                part_label1.destroy()
            # Test label2
            try:
                part_label2
            except NameError:
                part_label2 = None

            if part_label2 is not None:
                part_label2.destroy()
            # Test label3
            try:
                part_label3
            except NameError:
                part_label3 = None

            if part_label3 is not None:
                part_label3.destroy()

            # Display answer
            part_label1 = Label(app, text='Yes', font=('bold',
14), bg='#facd54', pady=20)
            part_label1.place(x=400, y=105)
        else:
            # Test label1
            try:
                part_label1
            except NameError:
                part_label1 = None

            if part_label1 is not None:
                part_label1.destroy()
            # Test label2
            try:
                part_label2
            except NameError:
                part_label2 = None

            if part_label2 is not None:
                part_label2.destroy()
            # Test label3
            try:
                part_label3
            except NameError:
                part_label3 = None

            if part_label3 is not None:
                part_label3.destroy()
```

```
            # Display answer
            part_label2 = Label(app, text='No', font=('bold',
14), bg='#facd54', pady=20)
            part_label2.place(x=400, y=105)
        filePath = "unknow"
    else:
        # Test label1
        try:
            part_label1
        except NameError:
            part_label1 = None

        if part_label1 is not None:
            part_label1.destroy()
        # Test label2
        try:
            part_label2
        except NameError:
            part_label2 = None

        if part_label2 is not None:
            part_label2.destroy()
        #Test label3
        try:
            part_label3
        except NameError:
            part_label3 = None

        if part_label3 is not None:
            part_label3.destroy()

        # Display answer
        part_label3 = Label(app, text=errmsg, font=('bold', 14),
bg='#facd54', pady=20)
        part_label3.place(x=400, y=105)

        print(errmsg)
        return errmsg
    if os.path.exists("recordingAudio.wav"):
        os.remove("recordingAudio.wav")


# Create Window
app = Tk()
app.resizable(False, False)

# background
filename = PhotoImage(file="img/bn3-bg.png")
background_label = Label(app, image=filename)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

# Bouton Choose File
add_btn = Button(app, text='Choose File', width=12,
command=chooseFile)
add_btn.place(x=200, y=200)

# Bouton Detect
add_btn = Button(app, text='Detect', width=12, command=execAI)
```

```
add_btn.place(x=100, y=200)

# Label
part_label = Label(app, text='Do you have Parkinson ?',
bg='#facd54', font=('bold', 14), pady=20)
part_label.place(x=185, y=105)

if os.path.exists("recordingAudio.wav"):
    os.remove("recordingAudio.wav")

# App title
app.title('AI Parkinson Detector')
app.geometry('600x400')
App(app)
app.mainloop()
```

3 Works Cited

[1] V. L. D. T. M. D. Joseph M. Savitt, “Diagnosis and treatment of Parkinson disease: molecules to medicine,” *Science in medicine,* pp. 1744-1754.

[2] Y. C.-R. C. J. P. David Montaña, “A Diadochokinesis-based expert system considering articulatory features of plosive consonants for early detection of Parkinson’s,” *Computer Methods and Programs in Biomedicine,* pp. 89-97, 2017.

[3] R. G. S. S. Sonia Mittal, “Model for Vocal Tract Filter,” *International Journal of Computer Science and Communication Engineering,* pp. 101-106, 2012.

[4] d. R. C. J. Rusz, “Quantitative acoustic measurements for characterization of speech and voice disorders in early untreated Parkinson’s,” *Czech Technical University in Prague,* pp. 350-368, 2010.

[5] P. o. P. D. U. N. C. w. D. T. U. G. Dynamics. [Online]. Available: https://dl.acm.org/citation.cfm?id=3168785.

[6] B. E. Sakar, “Collection and Analysis of a Parkinson Speech Dataset With Multiple Types of Sound Recordings,” [Online]. Available: https://www.researchgate.net/publication/260662600_Collection_and_Analysis_of_a_Parkinson_Speech_Dataset_With_Multiple_Types_of_Sound_Recordings.

[7] T. Bocklet, “Automatic Evaluation of Parkinson’s Speech - Acoustic, Prosodic and Voice,” [Online]. Available: http://www5.informatik.uni-erlangen.de/Forschung/Publikationen/2013/Bocklet13-AEO.pdf.

[8] J. Rusz, “Quantitative acoustic measurements for characterization of speech and voice disorders in early untreated Parkinson’s diseas,” 2011. [Online]. Available: https://asa.scitation.org/doi/pdf/10.1121/1.3514381?class=pdf.

[9] D. R. G. Ramani, “Parkinson Disease Classification using Data Mining Algorithms,” [Online]. Available: https://pdfs.semanticscholar.org/e89b/cbbc3dca714b0e1be724f7a92126311b1f3c.pdf.

[10] S. o. M. U. Department of Plastic Surgery, “Physiology of Speech Production,” [Online]. Available: https://link.springer.com/chapter/10.1007/978-4-431-68358-2_2.

[11] A. d. o. P. d. i. r. s. s. i. t. d. languages, 2016. [Online]. Available: https://asa.scitation.org/doi/10.1121/1.4939739.

[12] “Parkinson's disease - Mayo Clinic,” , . [Online]. Available: http://www.mayoclinic.org/diseases-conditions/parkinsons-disease/basics/definition/con-20028488. [Accessed 15 12 2019].

[13] Q. T.F, “Discrete Time Speech Signal Processing: Principles and Practice,” in *Production and Classification of Speech Sounds*, Prentice Hall, 2006, pp. 55-110.

[14] J. C. V´asquez-Correa, “Speech, Automatic Detection of Parkinson’s Disease from Continuous Speech Recorded in Non-Controlled Noise Conditions,” 2015.

[15] E. Belalcazar-Bola, “Automatic Detection of Parkinson’s Disease Using”.

[16] A. N. I. B. Y. S. a. R. E. Y. Jashmin K. Shah, “ROBUST VOICED/UNVOICED CLASSIFICATION USING NOVEL FEATURES AND GAUSSIAN MIXTURE MODEL”.

[17] A. Ng, “Support Vector Machines,” [Online]. Available: http://cs229.stanford.edu/notes/cs229-notes3.pdf.

[18] T. Fletcher, “Support Vector Machines Explained,” 2008. [Online]. Available: https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf.

[19] m. K. U. Kumar, “classification of parkinsons disease using multipass Lvq, Logistic Model Tree, K-star for Audio Data set,” 2011. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:519092/FULLTEXT01.pdf.