LOYOLA INSTITUTE OF TECHNOLOGY,CHENNAI

ANNA UNIVERSITY::CHENNAI – 600025

PROJECT TITLE :DETECTING PARKINSON'S DISEASE USING MACHINE LEARNING

A NALAIYA THIRAN PROJECT REPORT

SUBMITTED BY

TEAM SIZE : 4

TEAM LEADER : ANCY A (210919205004)

TEAM MEMBER : MURUGANANTHI S (210919205037)

TEAM MEMBER : REJI PRINCY A (210919205046)

TEAM MEMBER : SUVATHI M (210919205056)

TEAM ID : PNT2022TMID08567

MENTOR NAME: Mrs .FATHIMA **EVALUATOR NAME : Mr.D PRABHU DURAI**

TABLE OF CONTENT

ABSTRACT

Parkinson's disease is one of the supreme neurodegenerative problems of the human vital nervous organism. Parkinson's disease (PD) is a progressive neurological disorder commonly presented with tremors, slowness of movement, gait and balance issues. Additional problems could be speech, sensory disturbances, sleep issues, cognitive decline and psychological issues. Most of this will directly affect the day-to- day activities of individuals and result in reduced independence, which might lead to social isolation. It is a matter of sorrow that no specific clinical tests were introduced to detect Parkinson's disease correctly. As Parkinson's disease is non-communicable, early- stage detection of Parkinson's can prevent further damage in humans suffering from it.

LIST OF KEYWORDS :

PD(Parkinson Disease)

SVM(Suport Vector Machine)

ANN(Artificial Neural Network)

KNN(K Nearest Neighber)

DS(Data Science)

Machine Learning

CHAPTER 1

INTRODUCTION

Parkinson's disease is one of the supreme neurodegenerative problems of thehuman vital nervous organism. Parkinson's disease (PD) is a progressive neurological disorder commonly presented with tremors, slowness of movement, gait and balance issues. Additional problems could be speech, sensory disturbances, sleep issues, cognitive decline and psychological issues. Most of this will directly affect the day-to-day activities of individuals and result in reduced independence, which might lead to social isolation. It is amatter of sorrow that no specific clinical tests were introduced to detect Parkinson's disease correctly. As Parkinson's disease is non-communicable, early-stage detection of Parkinson's can prevent further damage in humans suffering from it.

OBJECTIVE

To understand the problem for to classify if it is a regression or a classification kind of problem.

- To pre-process the image by using different data pre-processing techniques.
- To implement the algorithm by using OpenCV framework and machine learning to automatically detect Parkinson’s disease in hand-drawn images of spirals and waves.
- To know how to find the accuracy of the model.
- To build web application using the Flask framework that features the detection of Parkinson's Disease.

1.2. LITERATURE SURVEY

PROJECT NAME : Parkinson Disease Using Machine Learning

AUTHOR NAME : Surekha Tadse , Muskan

DATE :14 Mar 2021

ABSTRACT:

Advance technology such as Data Science can be used to find solutions to medical science problems, by using its data and implementing machine learning Alogrithms on it, to draw the insights and patterns from the data and spotout the possibilities. The system has achieved a much better end up predicting the palladium patient is healthy or not, XGBoost provided the high accuracy of 96% and therefore theMatthews parametric statistic(MCC) of 89%.

PROJECT NAME: Early Detection of Parkinson’s Disease Using Deep Learning and Machine Learning

AUTHOR NAME :Wu Wang, Junho Lee ,Fouzi Harrou

DATE: Year 2020

ABSTRACT:

Accurately detecting Parkinson’s disease (PD) at an early stage is certainly indispensable for slowing down its progress and providing patients the possibility of accessing to disease modifying therapy. Towards this end, the premotor stage in PD should be carefully monitored. An innovative deep-learning technique is introduced to early uncover whether an individual is affected with PD or not based on premotor features.

PROJECT NAME: Parkinson's Disease Diagnosis Using Machine Learning and Voice

YEAR: June 2021

AUTHOR NAME: Timothy J. Wroge ,Yasin Ozkanca ,Cenk Demiroglu ,Dong Si

ABSTRACT:

Biomarkers derived from human voice can offer in-sight into neurological disorders, such as Parkinson's disease (PD), because of their underlying congnitive and neuromuscular function. PD is a progressive nerodgenerative disorder that affects about one million people in the united states, housand new clinical diagnoses made each year .

PROJECT TITLE : Parkinson’s Disease Detection based onChanges of Emotions during speech

YEAR :2018

AUTHOR NAME : Justyna Skibinska ,Radim Burge**t**

ABSTRACT :

Parkinson’s Disease Detection based on Changes of Emotions during speech Parkinson’s disease (PD) is theneurodgenerative disease which affects 2-3 % of the population beyond 65 years ofage in Eu. When PD treatment is administered early, it is significantly more effective.Unfortunately, it is quite challenging to detect this disease at its early stage and when the symptoms can be regonized it is usually quite late.

CHAPTER 2

1. PROBLEM STATEMENT

In this activity we are expected to prepare problem – Solution fit document and submit for review

DISADVANTAGE

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount o accuracy and relevancy. It also needs massive resources to function.This can mean additional requirements of computer power for you

3. PROPOSED SYSTEM

Parkinson's disease (PD) is a neurological disease that has progressed to an advanced stage. In the early stages of Parkinson's disease, roughly 90% of people with the disease Have speech problems. As a result, speech features were used To classify this condition in this study. Jitter, shimmer, basic Frequency parameters, harmonicity parameters, Recurrence Period Density Entropy (RPDE), Detrended Fluctuation Analysis (DFA), and Pitch Period Entropy are some of the most well-known speech aspects employed in PD research (PPE). Those characteristics were dubbed baseline characteristics in this study.

4. ADVANTAGE

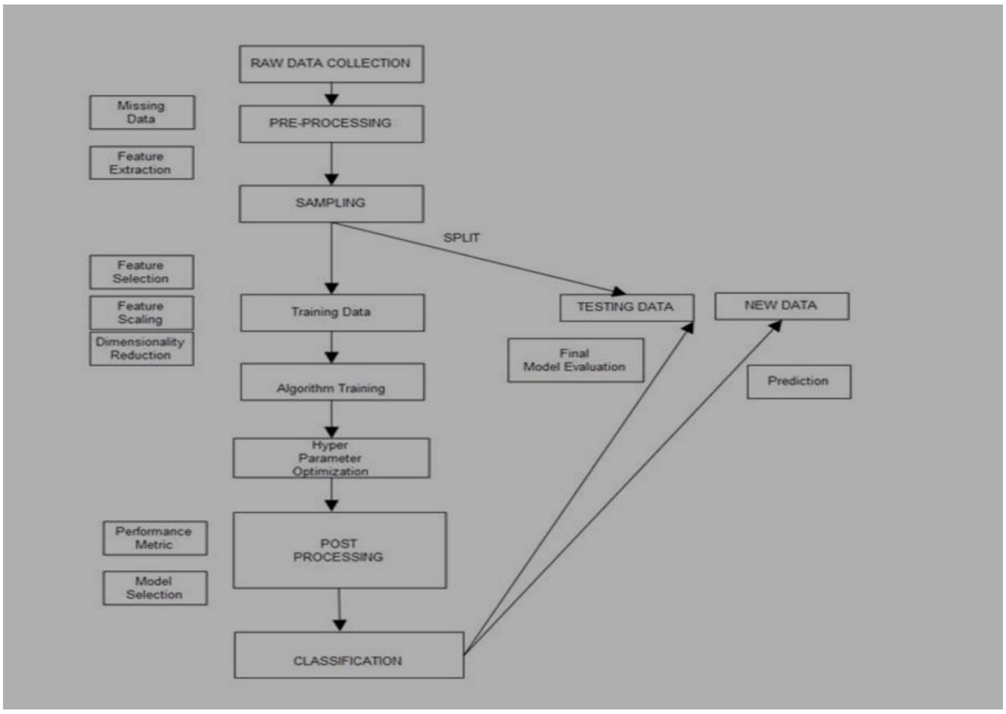
Even though high diagnostic accuracy of PD has been achieved in clinical settings, machine learning approaches have also reached high accuracy , while models including SVM and neural networks are particularly useful in (a) diagnosis of PD data modalities that have been overlooked in clinical decision making .

The use of machine learning models with feature selection techniques allows for assessing the relative importance of features of a large feature space in order to select the most differentiating ones, which is conventionally challenging using manual approaches.

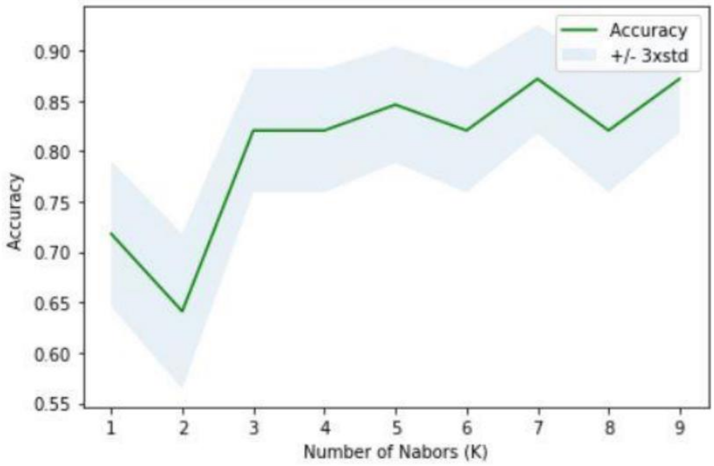
.

CHAPTER 3

3.1 STATEMENT ARCHITECTURE



3.2 TRAINING VS TESTING GRAPH



CHAPTER 4

4. SYSTEM REQUIREMENT

4.1 HARDWARE REQUIREMENT	
Hard Disk	500GB and Above
RAM	8GB and Above
Processor	I3 and Above

4.2 SOFTWARE REQUIREMENT	
Operating System	Windows 7 , 8, 10, 11 (64 bit)
Software	Anaconda, Jupyter Notebook, Python.
Tools or Framework	Numpy, Tensorflow, Seaborn, Keras, Pandas, Matplotlib

CHAPTER-5

SOURCE CODE

IBM CLOUD DEVELOPMENT

In [2]:

```
import warnings
warnings.filterwarnings("ignore") #Not to display the warnings

import numpy as np
import pandas as pd
import os, sys
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score #Modelmetrics
```

In [3]:

```
pip install lux
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: lux in /usr/local/lib/python3.7/dist-packages (0.5.1)
Requirement already satisfied: lux-widget in /usr/local/lib/python3.7/dist-packages (from lux) (0.1.11)
Requirement already satisfied: lux-api in /usr/local/lib/python3.7/dist-packages (from lux) (0.5.1)
Requirement already satisfied: scipy>=1.3.3 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (1.7.3)
Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (5.9.4)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (1.3.5)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (1.21.6)
Requirement already satisfied: iso3166 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (2.1.1)
Requirement already satisfied: autopep8>=1.5 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (2.0.0)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (1.0.2)
Requirement already satisfied: altair>=4.0.0 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (4.2.0)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (3.2.2)
Requirement already satisfied: sh in /usr/local/lib/python3.7/dist-packages (from lux-api->lux) (1.14.3)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.7/dist-packages (from altair>=4.0.0->lux-api->lux) (4.3.3)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.7/dist-packages (from altair>=4.0.0->lux-api->lux) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from altair>=4.0.0->lux-api->lux) (2.11.3)
Requirement already satisfied: toolz in /usr/local/lib/python3.7/dist-packages (from altair>=4.0.0->lux-api->lux) (0.12.0)
Requirement already satisfied: pycodestyle>=2.9.1 in /usr/local/lib/python3.7/dist-packages (from autopep8>=1.5->lux-api->lux) (2.9.1)
Requirement already satisfied: tomli in /usr/local/lib/python3.7/dist-packages (from autopep8>=1.5->lux-api->lux) (2.0.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from jsonschema>=3.0->altair>=4.0.0->lux-api->lux) (4.1.1)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema>=3.0->altair>=4.0.0->lux-api->lux) (22.1.0)
Requirement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema>=3.0->altair>=4.0.0->lux-api->lux)
(5.10.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema>=3.0->altair>=
4.0.0->lux-api->lux) (0.19.2)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from jsonschema>=3.0->altair>=4.0.0->lux-api->lux) (4.13.
0)
```

Data Preprocessing and Explotary Data Analysis(EDA)

In [4]:

```
parkinson_data = pd.read_csv('parkinsons.data')
print(parkinson_data)
```

```
               name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0    phon_R01_S01_1      119.992       157.302        74.997         0.00784   
1    phon_R01_S01_2      122.400       148.650       113.819         0.00968   
2    phon_R01_S01_3      116.682       131.111       111.555         0.01050   
3    phon_R01_S01_4      116.676       137.871       111.366         0.00997   
4    phon_R01_S01_5      116.014       141.781       110.655         0.01284   
..              ...          ...           ...           ...             ...   
190  phon_R01_S50_2      174.188       230.978        94.261         0.00459   
191  phon_R01_S50_3      209.516       253.017        89.488         0.00564   
192  phon_R01_S50_4      174.688       240.005        74.287         0.01360   
193  phon_R01_S50_5      198.764       396.961        74.904         0.00740   
194  phon_R01_S50_6      214.289       260.277        77.973         0.00567   

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  ...  \
0             0.00007   0.00370   0.00554     0.01109       0.04374  ...   
1             0.00008   0.00465   0.00696     0.01394       0.06134  ...   
2             0.00009   0.00544   0.00781     0.01633       0.05233  ...   
3             0.00009   0.00502   0.00698     0.01505       0.05492  ...   
4             0.00011   0.00655   0.00908     0.01966       0.06425  ...   
..                ...       ...       ...         ...           ...  ...   
190           0.00003   0.00263   0.00259     0.00790       0.04087  ...   
191           0.00003   0.00331   0.00292     0.00994       0.02751  ...   
192           0.00008   0.00624   0.00564     0.01873       0.02308  ...   
193           0.00004   0.00370   0.00390     0.01109       0.02296  ...   
194           0.00003   0.00295   0.00317     0.00885       0.01884  ...   

     Shimmer:DDA      NHR     HNR  status      RPDE       DFA   spread1  \
0        0.06545  0.02211  21.033       1  0.414783  0.815285 -4.813031   
1        0.09403  0.01929  19.085       1  0.458359  0.819521 -4.075192   
2        0.08270  0.01309  20.651       1  0.429895  0.825288 -4.443179   
3        0.08771  0.01353  20.644       1  0.434969  0.819235 -4.117501   
4        0.10470  0.01767  19.649       1  0.417356  0.823484 -3.747787   
..           ...      ...     ...     ...       ...       ...       ...   
190      0.07008  0.02764  19.517       0  0.448439  0.657899 -6.538586   
```

In [5]:

```python
parkinson_data
```

Out[5]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHI
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.0221
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.0192
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.0130
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.0135
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.0176
...	...	...	...	...	...	...	...	...	...	...	...	...	.
190	phon_R01_S50_2	174.188	230.978	94.261	0.00459	0.00003	0.00263	0.00259	0.00790	0.04087	...	0.07008	0.0276
191	phon_R01_S50_3	209.516	253.017	89.488	0.00564	0.00003	0.00331	0.00292	0.00994	0.02751	...	0.04812	0.0181
192	phon_R01_S50_4	174.688	240.005	74.287	0.01360	0.00008	0.00624	0.00564	0.01873	0.02308	...	0.03804	0.1071
193	phon_R01_S50_5	198.764	396.961	74.904	0.00740	0.00004	0.00370	0.00390	0.01109	0.02296	...	0.03794	0.0722
194	phon_R01_S50_6	214.289	260.277	77.973	0.00567	0.00003	0.00295	0.00317	0.00885	0.01884	...	0.03078	0.0439

95 rows × 24 columns

In [6]:

```python
parkinson_data.head(n=20)
```

Out[6]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHI
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.0221
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.0192
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.0130
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.0135
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.0176
5	phon_R01_S01_6	120.552	131.162	113.787	0.00968	0.00008	0.00463	0.00750	0.01388	0.04701	...	0.06985	0.0122

In [7]:

```
parkinson_data.tail(50)
```

Out[7]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR
5	phon_R01_S34_6	223.361	263.872	87.638	0.00352	0.00002	0.00169	0.00188	0.00506	0.02536	...	0.04137	0.01493
6	phon_R01_S35_1	169.774	191.759	151.451	0.01568	0.00009	0.00863	0.00946	0.02589	0.08143	...	0.11411	0.07530
7	phon_R01_S35_2	183.520	216.814	161.340	0.01466	0.00008	0.00849	0.00819	0.02546	0.06050	...	0.08595	0.06057
8	phon_R01_S35_3	188.620	216.302	165.982	0.01719	0.00009	0.00996	0.01027	0.02987	0.07118	...	0.10422	0.08069
9	phon_R01_S35_4	202.632	565.740	177.258	0.01627	0.00008	0.00919	0.00963	0.02756	0.07170	...	0.10546	0.07889
0	phon_R01_S35_5	186.695	211.961	149.442	0.01872	0.00010	0.01075	0.01154	0.03225	0.05830	...	0.08096	0.10952
1	phon_R01_S35_6	192.818	224.429	168.793	0.03107	0.00016	0.01800	0.01958	0.05401	0.11908	...	0.16942	0.21713
2	phon_R01_S35_7	198.116	233.099	174.478	0.02714	0.00014	0.01568	0.01699	0.04705	0.08684	...	0.12851	0.16265
3	phon_R01_S37_1	121.345	139.644	98.250	0.00684	0.00006	0.00388	0.00332	0.01164	0.02534	...	0.04019	0.04179
4	phon_R01_S37_2	119.100	128.442	88.833	0.00692	0.00006	0.00393	0.00300	0.01179	0.02682	...	0.04451	0.04611
5	phon_R01_S37_3	117.870	127.349	95.654	0.00647	0.00005	0.00356	0.00300	0.01067	0.03087	...	0.04977	0.02631
6	phon_R01_S37_4	122.336	142.369	94.794	0.00727	0.00006	0.00415	0.00339	0.01246	0.02293	...	0.03615	0.03191
7	phon_R01_S37_5	117.963	134.209	100.757	0.01813	0.00015	0.01117	0.00718	0.03351	0.04912	...	0.07830	0.10748
8	phon_R01_S37_6	126.144	154.284	97.543	0.00975	0.00008	0.00593	0.00454	0.01778	0.02852	...	0.04499	0.03828
9	phon_R01_S39_1	127.930	138.752	112.173	0.00605	0.00005	0.00321	0.00318	0.00962	0.03235	...	0.04079	0.02663
0	phon_R01_S39_2	114.238	124.393	77.022	0.00581	0.00005	0.00299	0.00316	0.00896	0.04009	...	0.04736	0.02073
1	phon_R01_S39_3	115.322	135.738	107.802	0.00619	0.00005	0.00352	0.00329	0.01057	0.03273	...	0.04933	0.02810
2	phon_R01_S39_4	114.554	126.778	91.121	0.00651	0.00006	0.00366	0.00340	0.01097	0.03658	...	0.05592	0.02707
3	phon_R01_S39_5	112.150	131.669	97.527	0.00519	0.00005	0.00291	0.00284	0.00873	0.01756	...	0.02902	0.01435
4	phon_R01_S39_6	102.273	142.830	85.902	0.00907	0.00009	0.00493	0.00461	0.01480	0.02814	...	0.04736	0.03882
5	phon_R01_S42_1	236.200	244.663	102.137	0.00277	0.00001	0.00154	0.00153	0.00462	0.02448	...	0.04231	0.00620

In [8]:
```
parkinson_data.shape
#(rows,columns)
```

Out[8]:
```
(195, 24)
```

In [9]:
```
#Capturing for null values if any of it is available
parkinson_data.isnull().sum()
```

Out[9]:
```
name                0
MDVP:Fo(Hz)         0
MDVP:Fhi(Hz)        0
MDVP:Flo(Hz)        0
MDVP:Jitter(%)      0
MDVP:Jitter(Abs)    0
MDVP:RAP            0
MDVP:PPQ            0
Jitter:DDP          0
MDVP:Shimmer        0
MDVP:Shimmer(dB)    0
Shimmer:APQ3        0
Shimmer:APQ5        0
MDVP:APQ            0
Shimmer:DDA         0
NHR                 0
HNR                 0
status              0
RPDE                0
DFA                 0
spread1             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

No null values are present in the data

In [10]:
```
parkinson_data.describe().round(2).style.background_gradient(cmap='Blues')
```

In [10]:

```
parkinson_data.describe().round(2).style.background_gradient(cmap='Blues')
```

Out[10]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)	Shimmer:APQ3	Shimr
nt	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	1
an	154.230000	197.100000	116.320000	0.010000	0.000000	0.000000	0.000000	0.010000	0.030000	0.280000	0.020000	
td	41.390000	91.490000	43.520000	0.000000	0.000000	0.000000	0.000000	0.010000	0.020000	0.190000	0.010000	
in	88.330000	102.140000	65.480000	0.000000	0.000000	0.000000	0.000000	0.000000	0.010000	0.080000	0.000000	
%	117.570000	134.860000	84.290000	0.000000	0.000000	0.000000	0.000000	0.000000	0.020000	0.150000	0.010000	
%	148.790000	175.830000	104.320000	0.000000	0.000000	0.000000	0.000000	0.010000	0.020000	0.220000	0.010000	
%	182.770000	224.210000	140.020000	0.010000	0.000000	0.000000	0.000000	0.010000	0.040000	0.350000	0.020000	
ax	260.100000	592.030000	239.170000	0.030000	0.000000	0.020000	0.020000	0.060000	0.120000	1.300000	0.060000	

In [11]:

```
parkinson_data.dtypes
```

Out[11]:

```
name                 object
MDVP:Fo(Hz)         float64
MDVP:Fhi(Hz)        float64
MDVP:Flo(Hz)        float64
MDVP:Jitter(%)      float64
MDVP:Jitter(Abs)    float64
MDVP:RAP            float64
MDVP:PPQ            float64
Jitter:DDP          float64
MDVP:Shimmer        float64
MDVP:Shimmer(dB)    float64
Shimmer:APQ3        float64
Shimmer:APQ5        float64
MDVP:APQ            float64
Shimmer:DDA         float64
NHR                 float64
HNR                 float64
status                int64
RPDE                float64
```

In [12]:

```python
for i in parkinson_data:
  print(i)
```

```
name
MDVP:Fo(Hz)
MDVP:Fhi(Hz)
MDVP:Flo(Hz)
MDVP:Jitter(%)
MDVP:Jitter(Abs)
MDVP:RAP
MDVP:PPQ
Jitter:DDP
MDVP:Shimmer
MDVP:Shimmer(dB)
Shimmer:APQ3
Shimmer:APQ5
MDVP:APQ
Shimmer:DDA
NHR
HNR
status
RPDE
DFA
spread1
spread2
D2
PPE
```

In [13]:

```python
#Verifying the unique values in the columns
for i in parkinson_data:
  print("##############################",i,"##############################")
  print()
  print(set(parkinson_data[i].tolist()))
```

```
############################## name ##############################
```

############################## name ##############################

{'phon_R01_S44_2', 'phon_R01_S02_3', 'phon_R01_S33_5', 'phon_R01_S10_5', 'phon_R01_S34_5', 'phon_R01_S44_4', 'phon_R01_S49_1', 'phon_R01_S16_6', 'phon_R01_S39_6', 'phon_R01_S21_3', 'phon_R01_S22_6', 'phon_R01_S39_3', 'phon_R01_S17_4', 'phon_R01_S35_3', 'phon_R01_S13_4', 'phon_R01_S21_2', 'phon_R01_S20_1', 'phon_R01_S43_6', 'phon_R01_S42_2', 'phon_R01_S27_1', 'phon_R01_S50_3', 'phon_R01_S33_1', 'phon_R01_S01_2', 'phon_R01_S06_2', 'phon_R01_S37_2', 'phon_R01_S24_3', 'phon_R01_S34_4', 'phon_R01_S20_4', 'phon_R01_S34_1', 'phon_R01_S04_2', 'phon_R01_S39_2', 'phon_R01_S04_1', 'phon_R01_S24_2', 'phon_R01_S13_6', 'phon_R01_S21_5', 'phon_R01_S13_3', 'phon_R01_S27_7', 'phon_R01_S49_5', 'phon_R01_S06_1', 'phon_R01_S27_4', 'phon_R01_S33_6', 'phon_R01_S27_3', 'phon_R01_S37_1', 'phon_R01_S50_2', 'phon_R01_S32_5', 'phon_R01_S37_3', 'phon_R01_S04_4', 'phon_R01_S27_2', 'phon_R01_S17_6', 'phon_R01_S01_6', 'phon_R01_S34_2', 'phon_R01_S04_5', 'phon_R01_S35_7', 'phon_R01_S50_5', 'phon_R01_S08_4', 'phon_R01_S26_3', 'phon_R01_S07_3', 'phon_R01_S08_2', 'phon_R01_S31_1', 'phon_R01_S26_1', 'phon_R01_S04_6', 'phon_R01_S05_2', 'phon_R01_S22_4', 'phon_R01_S24_4', 'phon_R01_S19_2', 'phon_R01_S06_5', 'phon_R01_S20_6', 'phon_R01_S42_5', 'phon_R01_S26_6', 'phon_R01_S13_1', 'phon_R01_S16_2', 'phon_R01_S22_5', 'phon_R01_S32_3', 'phon_R01_S10_1', 'phon_R01_S39_1', 'phon_R01_S20_2', 'phon_R01_S43_2', 'phon_R01_S44_5', 'phon_R01_S35_4', 'phon_R01_S01_1', 'phon_R01_S31_6', 'phon_R01_S33_2', 'phon_R01_S25_4', 'phon_R01_S50_6', 'phon_R01_S34_3', 'phon_R01_S26_4', 'phon_R01_S20_5', 'phon_R01_S18_3', 'phon_R01_S10_4', 'phon_R01_S22_1', 'phon_R01_S43_5', 'phon_R01_S13_5', 'phon_R01_S32_1', 'phon_R01_S49_2', 'phon_R01_S16_4', 'phon_R01_S35_2', 'phon_R01_S39_5', 'phon_R01_S31_2', 'phon_R01_S42_1', 'phon_R01_S25_5', 'phon_R01_S17_3', 'phon_R01_S25_2', 'phon_R01_S37_6', 'phon_R01_S02_4', 'phon_R01_S08_3', 'phon_R01_S24_6', 'phon_R01_S19_4', 'phon_R01_S07_1', 'phon_R01_S08_1', 'phon_R01_S44_1', 'phon_R01_S21_6', 'phon_R01_S44_3', 'phon_R01_S07_4', 'phon_R01_S18_6', 'phon_R01_S44_6', 'phon_R01_S35_1', 'phon_R01_S32_6', 'phon_R01_S19_3', 'phon_R01_S05_6', 'phon_R01_S31_4', 'phon_R01_S07_2', 'phon_R01_S50_1', 'phon_R01_S21_1', 'phon_R01_S06_4', 'phon_R01_S24_5', 'phon_R01_S18_4', 'phon_R01_S05_1', 'phon_R01_S18_5', 'phon_R01_S43_3', 'phon_R01_S10_2', 'phon_R01_S19_5', 'phon_R01_S22_3', 'phon_R01_S21_7', 'phon_R01_S16_5', 'phon_R01_S26_2', 'phon_R01_S17_1', 'phon_R01_S39_4', 'phon_R01_S34_6', 'phon_R01_S13_2', 'phon_R01_S22_2', 'phon_R01_S32_4', 'phon_R01_S16_1', 'phon_R01_S35_6', 'phon_R01_S18_1', 'phon_R01_S02_5', 'phon_R01_S37_5', 'phon_R01_S42_6', 'phon_R01_S05_5', 'phon_R01_S05_3', 'phon_R01_S08_6', 'phon_R01_S25_6', 'phon_R01_S31_3', 'phon_R01_S37_4', 'phon_R01_S25_3', 'phon_R01_S19_1', 'phon_R01_S20_3', 'phon_R01_S49_6', 'phon_R01_S16_3', 'phon_R01_S43_1', 'phon_R01_S01_3', 'phon_R01_S18_2', 'phon_R01_S06_3', 'phon_R01_S02_2', 'phon_R01_S33_4', 'phon_R01_S25_1', 'phon_R01_S31_5', 'phon_R01_S35_5', 'phon_R01_S26_5', 'phon_R01_S42_4', 'phon_R01_S21_4', 'phon_R01_S43_4', 'phon_R01_S17_5', 'phon_R01_S10_3', 'phon_R01_S32_2', 'phon_R01_S07_6', 'phon_R01_S10_6', 'phon_R01_S27_5', 'phon_R01_S27_6', 'phon_R01_S01_4', 'phon_R01_S04_3', 'phon_R01_S42_3', 'phon_R01_S02_1', 'phon_R01_S19_6', 'phon_R01_S49_3', 'phon_R01_S06_6', 'phon_R01_S02_6', 'phon_R01_S17_2', 'phon_R01_S01_5', 'phon_R01_S08_5', 'phon_R01_S33_3', 'phon_R01_S49_4', 'phon_R01_S07_5', 'phon_R01_S05_4', 'phon_R01_S24_1', 'phon_R01_S50_4'}
############################## MDVP:Fo(Hz) ##############################

{102.273, 110.568, 110.453, 110.739, 112.239, 112.15, 112.547, 113.4, 113.166, 113.715, 114.238, 114.554, 114.563, 115.322, 115.38, 116.879, 116.15, 116.388, 116.848, 116.286, 117.274, 117.87, 117.963, 117.004, 117.226, 118.747, 119.031, 88.333, 119.056, 119.1, 91.904, 120.078, 120.289, 120.256, 95.056, 95.73, 95.385, 96.106, 95.605, 100.77, 100.96, 98.804, 121.345, 104.4, 122.336, 106.516, 107.332, 108.807, 109.86, 110.793, 110.707, 112.014, 112.876, 114.847, 110.417, 116.676, 116.014, 116.682, 119.992, 120.267, 120.08, 122.188, 122.964, 124.445, 120.552, 122.4, 126.344, 128.001, 129.336, 125.036, 125.791, 126.512, 125.641, 128.451, 128.94, 136.926, 136.969, 136.358, 139.173, 140.341, 139.224, 142.167, 143.533, 144.188, 142.729, 146.845, 138.19, 148.09, 148.272, 150.258, 151.955, 152.845, 153.046, 153.848, 153.88, 156.405, 155.358, 152.125, 157.821, 157.447, 159.116, 162.568, 163.656, 155.078, 158.219, 166.605, 167.93, 168.778, 166.888, 170.756, 171.041, 170.368, 173.917, 173.898, 169.774, 176.17, 177.876, 176.858, 178.222, 180.198, 180.978, 176.281, 179.711, 184.055, 178.285, 186.163, 187.733, 182.018, 138.145, 183.52, 188.62, 186.695, 193.03, 192.818, 197.076, 198.383, 199.228, 200.714, 201.464, 202.266, 203.184, 204.664, 198.458, 206.327, 202.805, 208.519, 209.144, 210.141, 208.083, 209.516, 214.289, 217.116, 145.174, 222.236, 223.365, 223.361, 228.832, 229.401, 228.969, 148.79, 148.143, 148.462, 236.2, 237.226, 149.689, 237.323, 240.301, 241.404, 242.852, 243.439, 244.99, 245.51, 150.44, 149.818, 151.884, 151.989, 151.872, 151.737, 252.455, 260.105, 154.003, 116.556, 156.239, 202.632, 174.188, 174.688, 176.824, 116.342, 126.144, 197.569, 198.116, 198.764, 201.774, 202.544, 127.93}
############################## MDVP:Fhi(Hz) ##############################

{206.008, 131.669, 211.961, 565.74, 126.778, 217.627, 217.552, 581.289, 116.443, 586.567, 588.518, 592.03, 119.167, 128.143, 102.145, 102.305, 107.715,

In [14]:
```
parkinson_data['PPE'].tolist()
```

Out[14]:
```
[0.284654,
 0.368674,
 0.332634,
 0.368975,
 0.410335,
 0.357775,
 0.211756,
 0.163755,
 0.231571,
 0.271362,
 0.24974,
 0.275931,
 0.138512,
 0.199889,
 0.1701,
 0.234589,
 0.218164,
 0.430788,
 0.377429,
 0.322111,
 0.365391,
 0.259765,
 0.285695,
 0.253556,
 0.215961,
 0.219514,
 0.147403,
 0.162999,
 0.108514,
 0.135242,
 0.085569,
 0.068501,
 0.09632,
 0.056141,
 0.044539,
 0.05761,
 0.165827,
 0.173218,
 0.141929,
```

In [15]:

```
variable=parkinson_data['status'].value_counts()
variable_data=pd.DataFrame({'status':variable.index,'values':variable.values})
variable_data
```

Out[15]:

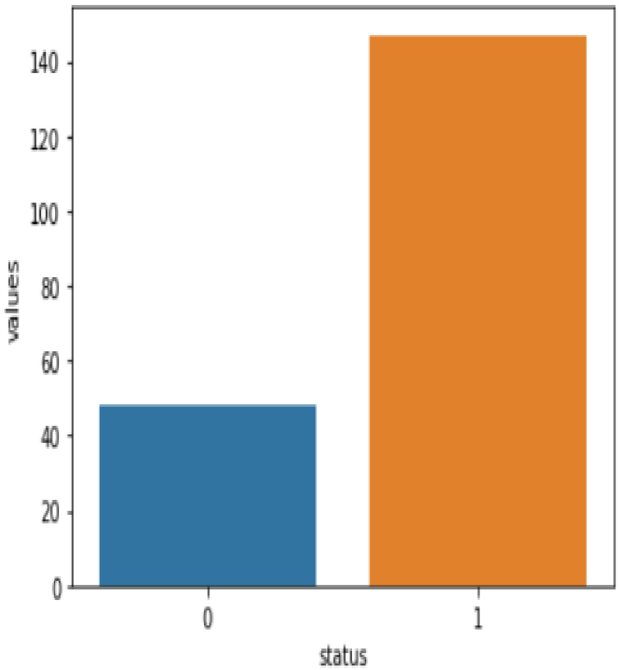
	status	values
0	1	147
1	0	48

Data visualization

In [16]:

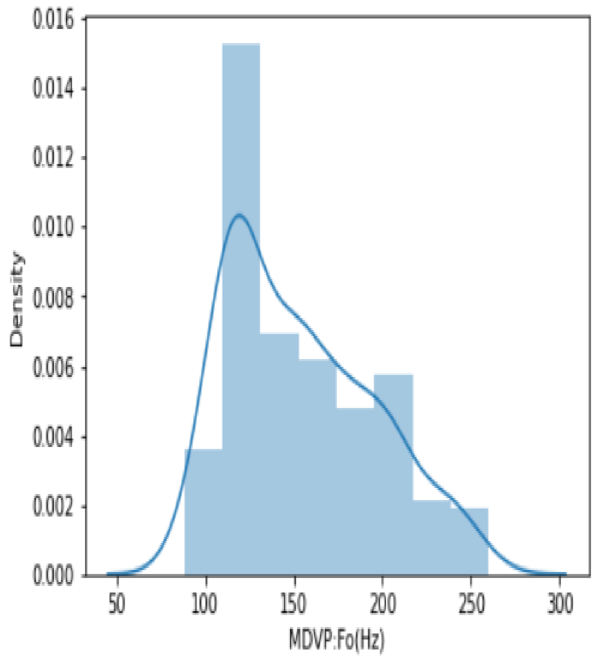
```
#Data visualization
import seaborn as sns
import matplotlib.pyplot as plt
variable = parkinson_data["status"].value_counts()
variable_data = pd.DataFrame({'status':variable.index,'values':variable.values})
sns.barplot(x='status',y='values',data=variable_data)
```

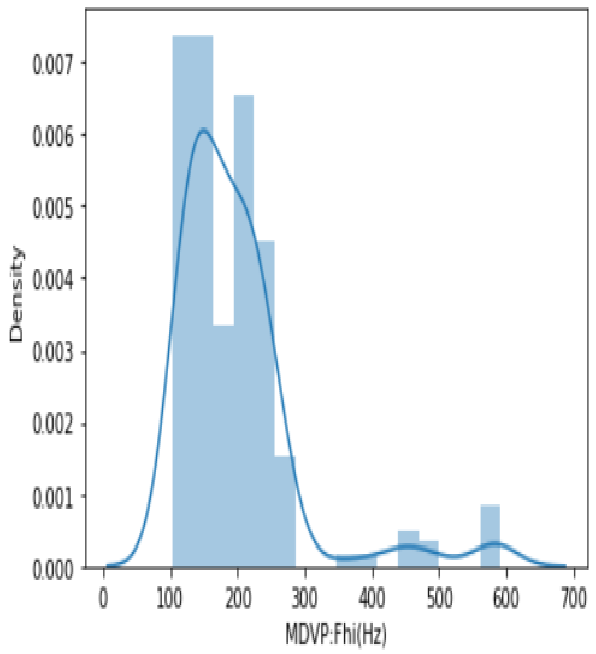
Out[16]:



In [17]:

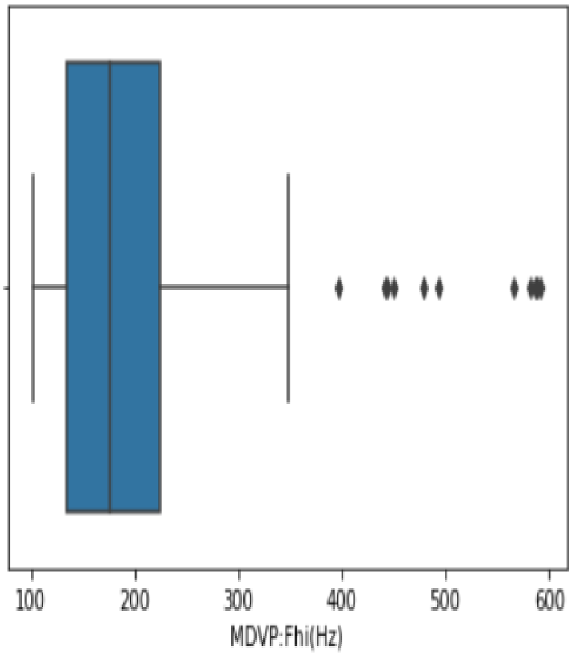
```
#Analyzing the distribution of the data using distplot
def distplots(col):
  sns.distplot(parkinson_data[col])
  plt.show()

for i in list(parkinson_data.columns)[1:]:
  distplots(i)
```

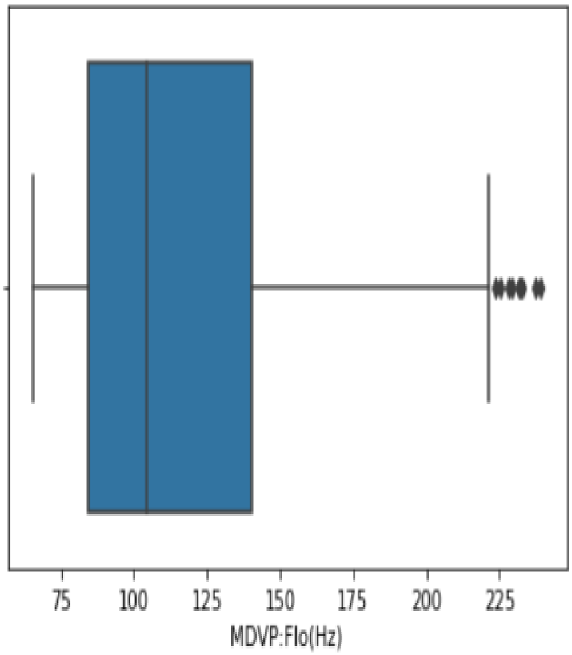




In [20]:

```
#Checking for outliers using boxplot from seabron framework across different quartiles
def boxplots(col):
  sns.boxplot(parkinson_data[col])
  plt.show()

for i in list(parkinson_data.select_dtypes(exclude=["object"]).columns)[1:]:
  boxplots(i)
```

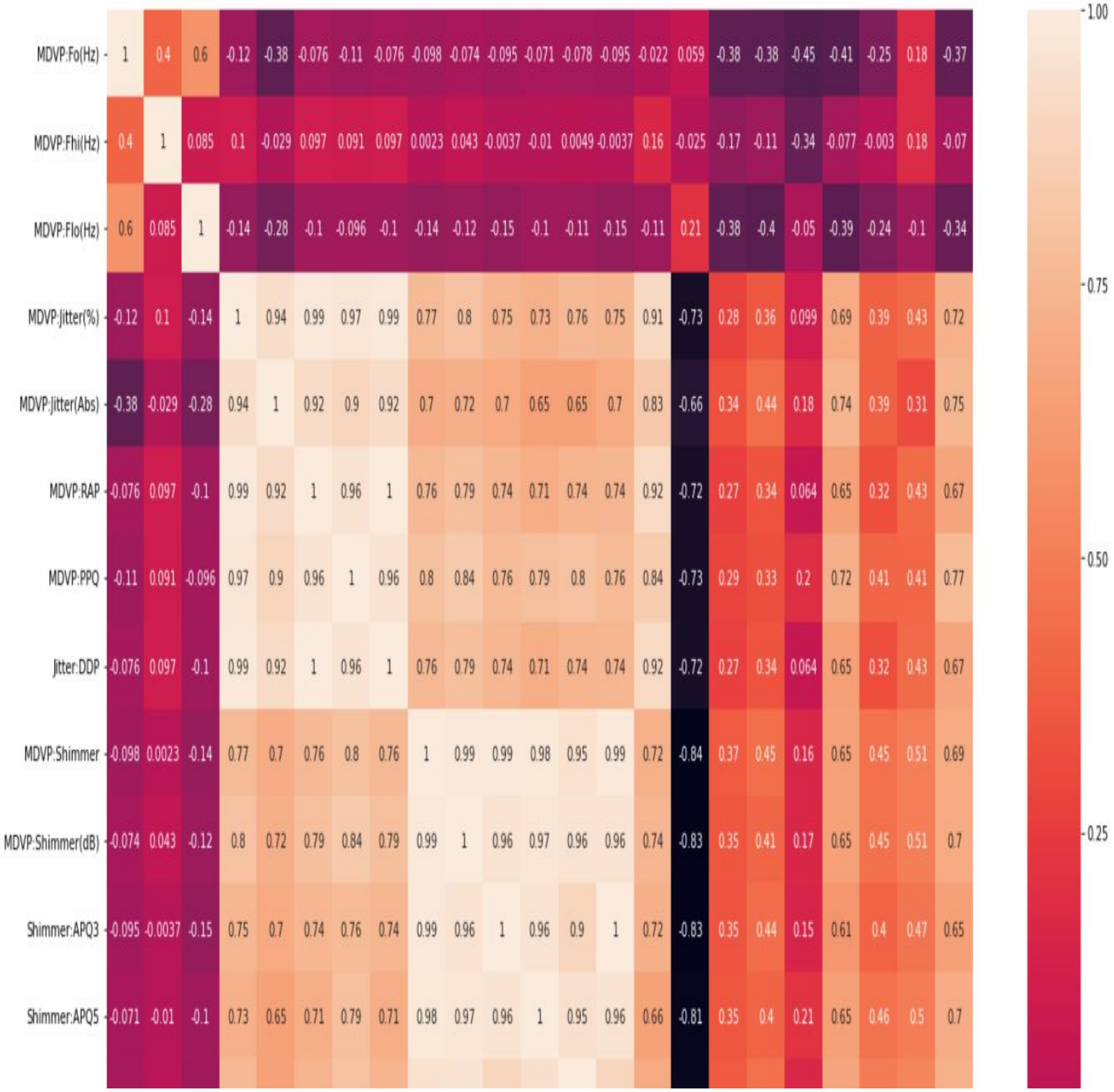




In [21]:

```
#Figuring out the correlations using heatmap to visualize between the features and patterns in the data used for this project

plt.figure(figsize=(20,20))
correlation_data=parkinson_data.corr()
sns.heatmap(correlation_data,annot=True)
```

Out[21]:



In [22]:

```
#We are making the final changes in the data by dividing the data into independent as x and dependent variables as y and removing the ID column
x = parkinson_data.drop(["status","name"],axis=1)
y = parkinson_data["status"]
#It is done to integrate the two x and y variables into the model building steps
```

In [23]:

```
#After the changes,let's detect the label balance
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter #For priortizing the importance to store elements as dictionary keys, and their counts as values.
print(Counter(y))
```

```
Counter({1: 147, 0: 48})
```

In [24]:

```
#Now,we are balancing the labels
ROS = RandomOverSampler() #To compensate the imbalance part present in the data
x_ROS,y_ROS = ROS.fit_resample(x, y)
print(Counter(y_ROS))
```

```
Counter({1: 147, 0: 147})
```

Scaling the data

In [25]:

```
#It is very much important to scale the data for the betterment of the model using such as Support Vector Machine and K Nearest Neighbor Algorithms
Scaler_data = MinMaxScaler((-1,1))
x = Scaler_data.fit_transform(x_ROS)
y = y_ROS
```

In [26]:

```
#Now, we are applying feature engineering and Principle Component Analysis using Data Mining for extracting high variance features and transforms
#Mining value from the data
#We are choosing the minimal number of variance as 0.95 as to target that the 95% of the variance is proved or confined from the mining process

from sklearn.decomposition import PCA
Princple_CA = PCA(.95)
X_PCA = Princple_CA.fit_transform(x)
print(x.shape)
print(X_PCA.shape)
```

```
(294, 22)
(294, 8)
```

We have noticed that eight columns are needed to prove the 95 % of the data is retained

In [27]:

```
#Here the Parkinson_data is splitted into training and testing sets by maintaining 20% of the data sample for testing step
x_train,x_test,y_train,y_test = train_test_split(X_PCA,y, test_size=0.2, random_state=7)
```

Since the labels from the data has been balanced so we are to use metrics such as accuracy_score, confusion_matrix, f1_score, precision_score and recall_score

Since we need to get boolean responses after the disease prediction so we are using Logistic Regression by the use of independent variables by assuming that the parkinson_data is linearably separable

Model Building (Training and Testing)

Data mining and performance metrics

In [28]:

```
#We are going to import and use it for assessing the model using performance metrics from Classification process
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score
List_metrics = []
List_accuracy = []

#Logistic Regression
from sklearn.linear_model import LogisticRegression
Classification_model = LogisticRegression(C=0.4,max_iter=1000,solver='liblinear')
Log_Regression = Classification_model.fit(x_train, y_train)
y_pred = Classification_model.predict(x_test)  #Prediction
Log_Regression_accuracy = accuracy_score(y_test, y_pred)  #Accuracy
print("The accuracy score with Logistic regression is:",Log_Regression_accuracy)

#Decision Tree Classificaton using supervised machine learning for classifiying the data with confident accuracy
from sklearn.tree import DecisionTreeClassifier
Classification_tree = DecisionTreeClassifier(random_state=14)
Decision_tree = Classification_tree.fit(x_train, y_train)
y_pred2 = Classification_tree.predict(x_test) #Prediction
Dec_tree_accuracy = accuracy_score(y_test, y_pred2) #Accuracy
print("The accuracy score with Decision Tree Classifier is:",Dec_tree_accuracy)

#Random Forest Classifier is used for its high dimensionality and accuracy capabilities, here information gain is priortized
from sklearn.ensemble import RandomForestClassifier
Classification_random = RandomForestClassifier(random_state=14)
RFE = Classification_random.fit(x_train, y_train)
y_pred3 = Classification_random.predict(x_test) #Prediction
Ran_For_accuracy = accuracy_score(y_test, y_pred3) #Accuracy
print("The accuracy score with Random Forest Classifier(Information gain) is:",Ran_For_accuracy)

#Random Forest Classifier with entropy condition
from sklearn.ensemble import RandomForestClassifier
Classification_entropy = RandomForestClassifier(criterion='entropy')
RFE = Classification_entropy.fit(x_train,y_train)
y_pred4 = Classification_entropy.predict(x_test)
Random = accuracy_score(y_test, y_pred4)
print("The accuracy score with Random Forest Classifier(Entropy) is:",Random)

#Using Support Vector Machine (SVM) for to enhance the similarity and to increase the scaling factor of the model
```

```python
#Random Forest Classifier with entropy condition
from sklearn.ensemble import RandomForestClassifier
Classification_entropy = RandomForestClassifier(criterion='entropy')
RFE = Classification_entropy.fit(x_train,y_train)
y_pred4 = Classification_entropy.predict(x_test)
Random = accuracy_score(y_test, y_pred4)
print("The accuracy score with Random Forest Classifier(Entropy) is:",Random)

#Using Support Vector Machine (SVM) for to enhance the similarity and to increase the scaling factor of the model
from sklearn.svm import SVC
Parkinson_model = SVC(cache_size=100)
Support_vector_machine = Parkinson_model.fit(x_train, y_train)
y_pred5 = Parkinson_model.predict(x_test)
Support_accuracy = accuracy_score(y_test, y_pred5)
print("The accuracy score with Support Vector Machine is:",Support_accuracy)

#K Nearest Neighbor Classifier for better effectiveness
from sklearn.neighbors import KNeighborsClassifier
KNN_parkinson = KNeighborsClassifier(n_neighbors=3)
K_Nearest_Neighbor_Classifier = KNN_parkinson.fit(x_train, y_train)
KNN_predict = KNN_parkinson.predict(x_test)
KNN_accuracy = accuracy_score(y_test, KNN_predict)
print("The accuracy score with K Nearest Neighbor Algorithm is:",KNN_accuracy)

#GaussianNB
from sklearn.naive_bayes import GaussianNB
GNB = GaussianNB()
Model_NB = GNB.fit(x_train,y_train)
pred_gnb = Model_NB.predict(x_test)
GNB_accuracy = accuracy_score(y_test, pred_gnb)
print("The accuracy score with Gaussian Naive Bayes is:",GNB_accuracy)

print("\nLet's see the overall accuracy of the built model that is been created below, view the overall accuracy score below!")
Overall_accuracy_percentage = Log_Regression_accuracy+Dec_tree_accuracy+Ran_For_accuracy+Random+Support_accuracy+KNN_accuracy+GNB_accuracy
Average_accuracy = (Overall_accuracy_percentage)/7
print("The accuracy of all the combined metrics for the model is:",Average_accuracy/0.01)
```

```
The accuracy score with Logistic regression is: 0.847457627118644
The accuracy score with Decision Tree Classifier is: 1.0
The accuracy score with Random Forest Classifier(Information gain) is: 1.0
The accuracy score with Random Forest Classifier(Entropy) is: 1.0
The accuracy score with Support Vector Machine is: 0.9322033898305084
The accuracy score with K Nearest Neighbor Algorithm is: 0.9830508474576272
The accuracy score with Gaussian Naive Bayes is: 0.864406779661017
```

In [29]:
```
from sklearn.ensemble import VotingClassifier
VC = VotingClassifier(estimators=[('Classification_model',Classification_model),('Classification_tree',Classification_tree),('Classification_random',C
Model_VC = VC.fit(x_train, y_train)
Model_prediction = VC.predict(x_test)
Model_accuracy = accuracy_score(y_test,pred_gnb)
print(Model_accuracy)
```

```
0.864406779661017
```

XGBClassification - Supervised Machine Learning

In [30]:
```
Model_XG = XGBClassifier(random_state=0)
Model_XG.fit(x_train,y_train)
```

Out[30]:
```
XGBClassifier()
```

Assessing the model using metrics

In [31]:
```
y_predict = Model_XG.predict(x_test)
print(accuracy_score(y_test,y_predict)*100)
```

```
98.30508474576271
```

Hence by reducing the overfitting using XGBoost Classifier, we are getting accuracy_score of 98.30% for the model

Confusion metrics

In [32]:
```
from sklearn.metrics import confusion_matrix
ypre = Classification_model.predict(x_test)
ypre = (ypre>0.5)
confusion_matrix(y_test,ypre)
```

Out[32]:
```
array([[22,  2],
```

F1 score

In [33]:
```
from sklearn.metrics import f1_score
Variation_score = f1_score(y_test, Model_XG.predict(x_test), average='binary')
print(Variation_score/0.01)
```

```
98.55072463768116
```

Classification report

In [34]:
```
from sklearn import metrics
from sklearn.metrics import classification_report
print("\n Classification report for Model  %s:\n%s\n" % (Model_XG, metrics.classification_report(y_test, y_pred)))
```

```
 Classification report for Model  XGBClassifier():
              precision    recall  f1-score   support

           0       0.76      0.92      0.83        24
           1       0.93      0.80      0.86        35

    accuracy                           0.85        59
   macro avg       0.85      0.86      0.85        59
weighted avg       0.86      0.85      0.85        59
```

In [35]:
```
final_data = parkinson_data.rename(columns = {'MDVP:Fo(Hz)':'Fo','MDVP:Fhi(Hz)':'Fhi','MDVP:Flo(Hz)':'Flo','MDVP:Shimmer(dB)':'Shimmer'})
final_data
```

Out[35]:

	name	Fo	Fhi	Flo	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1

Saving the model

In [36]:
```
import pickle

with open( 'Parkinson_MLmodel.sav', 'wb') as f:
    pickle.dump(Model_XG,f)

with open('standardScalar.sav', 'wb') as f:
    pickle.dump(Scaler_data,f)
```

Deployment initiation process

In [37]:
```
!pip install -U ibm-watson-machine-learning
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ibm-watson-machine-learning
  Downloading ibm_watson_machine_learning-1.0.257-py3-none-any.whl (1.8 MB)
     |████████████████████████████████| 1.8 MB 6.4 MB/s
Collecting ibm-cos-sdk==2.7.*
  Downloading ibm-cos-sdk-2.7.0.tar.gz (51 kB)
     |████████████████████████████████| 51 kB 782 kB/s
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from ibm-watson-machine-learning) (2022.9.24)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from ibm-watson-machine-learning) (2.23.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from ibm-watson-machine-learning) (4.13.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from ibm-watson-machine-learning) (1.24.3)
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from ibm-watson-machine-learning) (0.8.10)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from ibm-watson-machine-learning) (21.3)
Collecting lomond
  Downloading lomond-0.3.3-py2.py3-none-any.whl (35 kB)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /usr/local/lib/python3.7/dist-packages (from ibm-watson-machine-learning) (1.3.5)
Collecting ibm-cos-sdk-core==2.7.0
  Downloading ibm-cos-sdk-core-2.7.0.tar.gz (824 kB)
     |████████████████████████████████| 824 kB 53.9 MB/s
Collecting ibm-cos-sdk-s3transfer==2.7.0
  Downloading ibm-cos-sdk-s3transfer-2.7.0.tar.gz (133 kB)
     |████████████████████████████████| 133 kB 63.6 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting docutils<0.16,>=0.10
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)
     |████████████████████████████████| 547 kB 30.1 MB/s
```

```python
from ibm_watson_machine_learning import APIClient
import json
```

```python
#Authenticate and set space
wml_credentials = {
    "apikey":"s3nNigNL1Ev3RNdHNux58n0UNRXQdCr4AzYDUmYrPwTV",
    "url":"https://us-south.ml.cloud.ibm.com/"
}
```

```python
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
```

```python
SPACE_ID=""
```

```python
wml_client.set.default_space(SPACE_ID)
#Output='SUCCESS'
```

Deploying the model

In [43]:
```
DEPLOMENT_MODEL_NAME1 = '/content/Parkinson_MLmodel.sav'
DEPLOYMENT_MODEL_NAME2 = '/content/standardScalar.sav'
BEST_MODEL = Model_XG
```

In [ ]:
```
software_spec_uid = wml_client.software_specifications.get_id_by_name('default_py3.7')

# Setup model meta
model_props = {
    wml_client.repository.ModelMetaNames.NAME: DEPLOYMENT_MODEL_NAME1,
    wml_client.repository.ModelMetaNames.NAME: DEPLOYMENT_MODEL_NAME2,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_0.23',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}

#Save model
model_details = wml_client.repository.store_model(
    model1=DEPLOYMENT_MODEL_NAME1,
    model2=DEPLOYMENT_MODEL_NAME2,
    meta_props=model_props,
    training_data=X_train.head(),
    training_target=y_train.head()
)
```

In [ ]:
```
model_details
```

In [ ]:
```
model_uid = wml_client.repository.get_model_uid(model_details); model_uid
```

```python
model_uid = wml_client.repository.get_model_uid(model_details); model_uid
```

```python
deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}

# Deploy
deployment = wml_client.deployments.create(
    artifact_uid=model_uid,
    meta_props=deployment_props
)

# Model deployment output result
deployment
```

DEVELOPING HTML PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"></script>
    <style>
       body {
                 background-image: url({{url_for('static',filename='/images/homebg3.jpg')}});
                 background-repeat: no-repeat;
                 background-attachment: fixed;
                 background-size: 100% 100%;
                 color: white;
}
    </style>
</head>
<body>
    <div class="container-fluid" style=
                                   "background-color:rgb(41, 41, 41);
                                   border-radius: 0px;">
        <ul class="nav justify-content-end">
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('home_page')}}"><b>Home</b></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('info_page')}}"><b>Info</b></a>
            </li>
            <li class="nav-item">
```

```
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('predict_page')}}"><b>Predict</b></a>
            </li>
          </ul>
    </div>

<div calss="container">

    <h1 style="text-align: center; margin-top: 10px;">
        <b>Detecting Parkinson's disease using Machine Learning</b></h1>
    <p style="font-size:30px;
        padding-left: 100px;
        padding-top: 80px;">Parkinson's disease is a progressive disorder
        that affects the nervous system and the parts of the body
        controlled by the nerves. Symptoms start slowly. The first
        symptom may be a barely noticeable tremor in just one hand.
        Tremors are common, but the disorder may also cause stiffness
        or slowing of movement.
        In the early stages of Parkinson's disease, your face may show
        little or no expression. Your arms may not swing when you walk.
         Your speech may become soft or slurred. Parkinson's disease
         symptoms worsen as your condition progresses over time.
        Although Parkinson's disease can't be cured, medications
        might significantly improve your symptoms. Occasionally,
        your health care provider may suggest surgery to regulate
        certain regions of your brain and improve your symptoms.
    </p>
</div>
</body>
</html>
```

```
{% extends "layout.html" %}
{% block content %}
<head>
<style>
        body {
                background-image: url({{url_for('static',filename='/images/predict.jpg')}});
                background-repeat: no-repeat;
                background-attachment: fixed;
                background-size: 100% 100%;
                color: white;
}
    </style>
</head>
    <div class="container">
    <form method="post" enctype='multipart/form-data'>
        <div class="mb-3">{{form.hidden_tag()}}</div>
        <div class="mb-3">{{form.filew.label(class="form-label fs-2")}}</div>
        <div class="mb-3">{{form.filew(class="form-control")}}</div>

        <div class="mb-3">{{form.files.label(class="form-label fs-2")}}</div>
        <div class="mb-3">{{form.files(class="form-control")}}</div>
        <div class="mb-3">{{form.submit(class="btn btn-primary")}}</div>
    </form>

    </div>
    <h1>{{res}}</h1>
{% endblock %}
```

```
{% extends "layout.html" %}
{% block content %}
<div calss="container">
    <h1 style="text-align: center; margin-top: 10px;">
        <b>Prevention is better than cure!</b></h1>
    <img src="{{url_for('static',filename='/images/prev.png')}}" style="width:20cm;margin-left:25%;">
</div>
{% endblock %}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body style="background-color:rgb(205, 205, 205)">
    <div class="container-fluid" style=
                                    "background-color:rgb(41, 41, 41);
                                    border-radius: 0px;">
        <ul class="nav justify-content-end">
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('home_page')}}"><b>Home</b></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('info_page')}}"><b>Info</b></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('predict_page')}}"><b>Predict</b></a>
            </li>
          </ul>
    </div>
    {% block content %}

    {% endblock %}
</body>
</html>
```

```
{% extends "layout.html" %}
{% block content %}
<head>
<div calss="container">
    <h1 style="text-align:center"><b>Prediction:Results</b></h1>
</div>
<style>
       body {
                background-image: url({{url_for('static',filename='/images/result.jpg')}});
                background-repeat: no-repeat;
                background-attachment: fixed;
                background-size: 100% 100%;
                color: white;
}
 </style>
 </head>

<div class="container text-center">
    <div class="row">
      <div class="col">
        <h1 style="margin-left: 6cm; margin-top: 30px;">Wave:
            <b>{{rew}}</b></h1>
        <img src="{{url_for('static',filename='/images/wave/wave.png')}}"
            style="margin-left:5cm;height:10cm;width:10cm">
      </div>
      <div class="col">
        <h1 style="margin-left: 6cm; margin-top: 30px;">Spiral:
            <b>{{res}}</b></h1>
        <img src="{{url_for('static',filename='/images/spiral/spiral.png')}}"
            style="margin-left:5cm;height:10cm;width:10cm">
      </div>
    </div>
  </div>
```

PYTHON CODE FOR APP

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


@app.route('/predict',methods=['GET','POST'])
def upload():
if request.method == "POST":
f=request.files['file']
basepath-os.path.dirname(_file_)storing the file directory
filepath-os.path.join(basepath, "uploads", f.filename) #storing the file
f.save(filepath)#saving the file


#Loading the saved model
print("[INFO] Loading model...")
model= pickle.loads (open('parkinson.pkt', "rb").read())
# pre-process the image in the same manner we did earlier
image cv2.imread(filepath)
output =image.copy()
#load the input image, convert it to grayscale, and resize
output = cv2.resize(output, (128, 128))
image cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
image = cv2.resize(image, (200, 200))
image = cv2.threshold (image, 0, 255,
cv2.THRESH BINARY INV | cv2.THRESH_OTSU)[1]


#quantify the image and make predictions based on the extracted
# features using the last trained Random Forest
features= feature.hog(image, orientations=9,
pixels_per_cell-(10, 10), cells_per_block-(2, 2),
```

```
pixels_per_cell-(10, 10), cells_per_block-(2, 2),
transform_sqrt=True, block_norm="L1")
preds = model.predict([features])
print(preds)
1s=["healthy", "parkinson"]
result =Is[preds[0]]



# draw the colored class label on the output image and add it
# the set of output images
color= (0, 255, 0) if result == "healthy" else (0, 0, 255)
cv2.putText(output,result, (3, 20), cv2.FONT HERSHEY SIMPLEX,0.5,color,2)
cv2.imshow("Output", output)
cv2.waitKey(0)
return result
return None
```

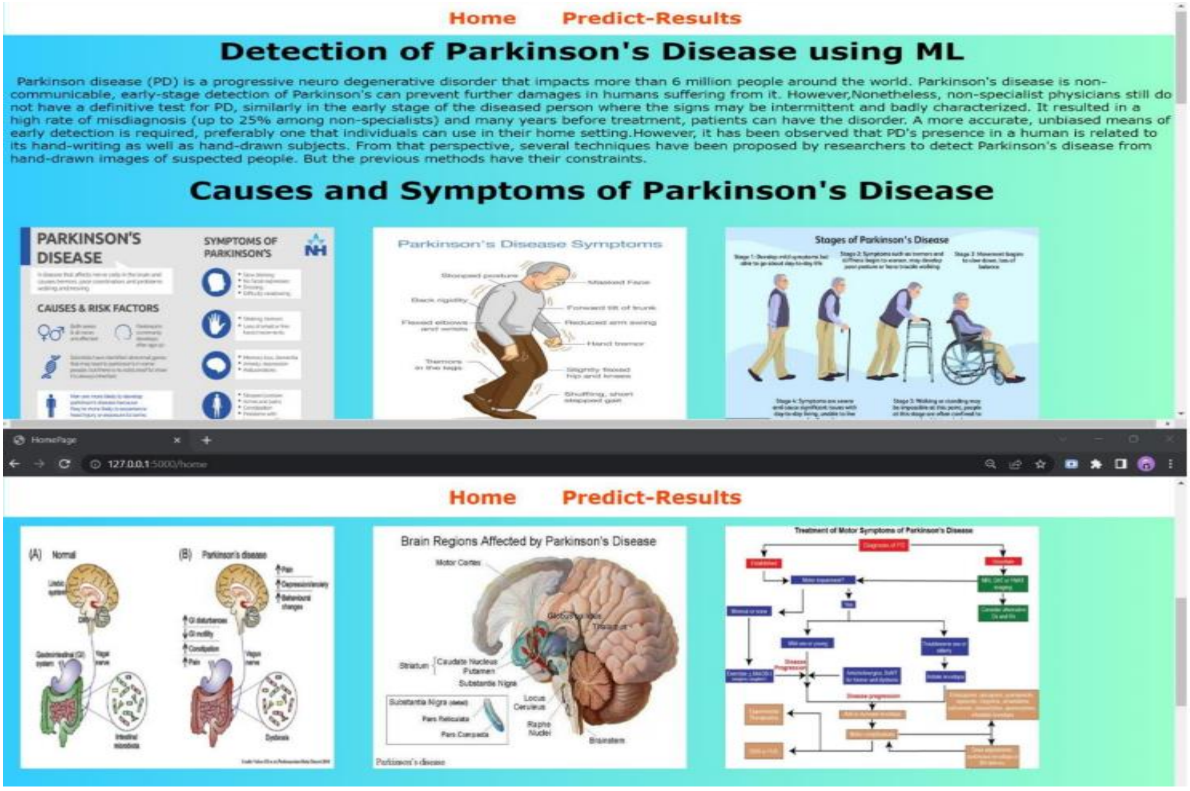
```python
# -*- coding: utf-8 -*-
"""app.py

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/11DIZF5Ec9mAEOz4ggwIxyqm53-zbrHoE
"""

rom flask import Flask, render_template, redirect, url_for
from Files import app
from flask_wtf import FlaskForm
from wtforms import FileField, SubmitField, validators
from werkzeug.utils import secure_filename
import os
import cv2
import pickle
from skimage import feature


class file_upload(FlaskForm):
    filew = FileField(label='Choose wave image', validators=[validators.DataRequired()
    files = FileField(label='Choose spiral image', validators=[validators.DataRequired
    submit = SubmitField(label='Predict')


@app.route('/')
def home_page():
    return render_template('home.html')


@app.route('/info')
def info_page():
    return render_template('info.html')
```

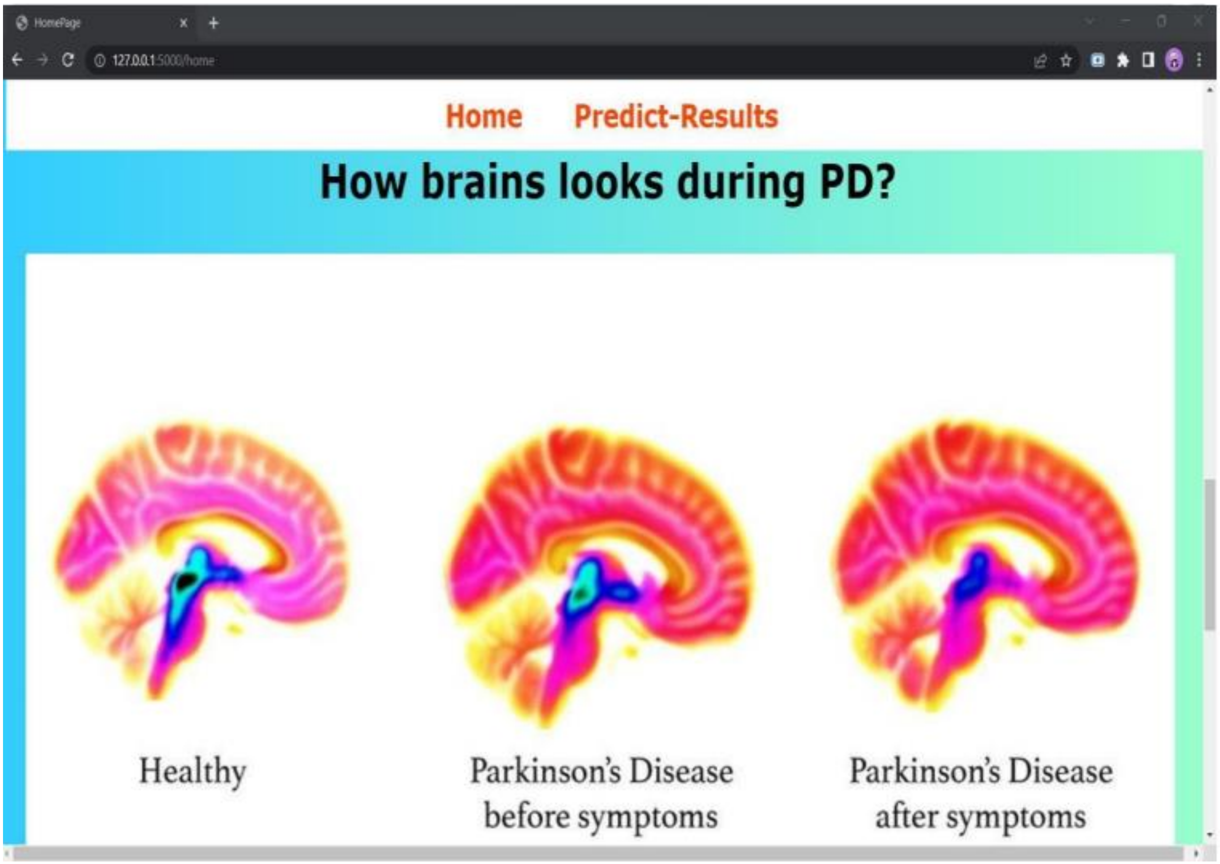
```
@app.route('/result', defaults={'results': "Something went wrong", 'resultw': "Something went wrong"}
@app.route('/result/<string:results>/<string:resultw>')
def result_page(results, resultw):
    return render_template('result.html', res=results, rew=resultw)


@app.route('/predict', methods=['GET', 'POST'])
def predict_page():
    form = file_upload()
    if form.validate_on_submit():
        filewave = form.filew.data
        filespiral = form.files.data
        filewave.save(os.path.join(os.path.abspath(os.path.dirname(__file__)),
                                   app.config['UPLOAD_FOLDER_WAVE'],
                                   secure_filename(filewave.filename)))
        filespiral.save(os.path.join(os.path.abspath(os.path.dirname(__file__)),
                                     app.config['UPLOAD_FOLDER_SPIRAL'],
                                     secure_filename(filespiral.filename)))

        modelw = pickle.loads(open('parkinsonWave.pkl', 'rb').read())
        models = pickle.loads(open('parkinsonSpiral.pkl', 'rb').read())

        def pre_wave(loc):
            file = os.listdir(loc)[0]
            img = os.path.join(loc, file)
            image = cv2.imread(img)
            output = image.copy()
            output = cv2.resize(output, (128, 128))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            image = cv2.resize(image, (200, 200))
            image = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

            features = feature.hog(image, orientations=9,
```

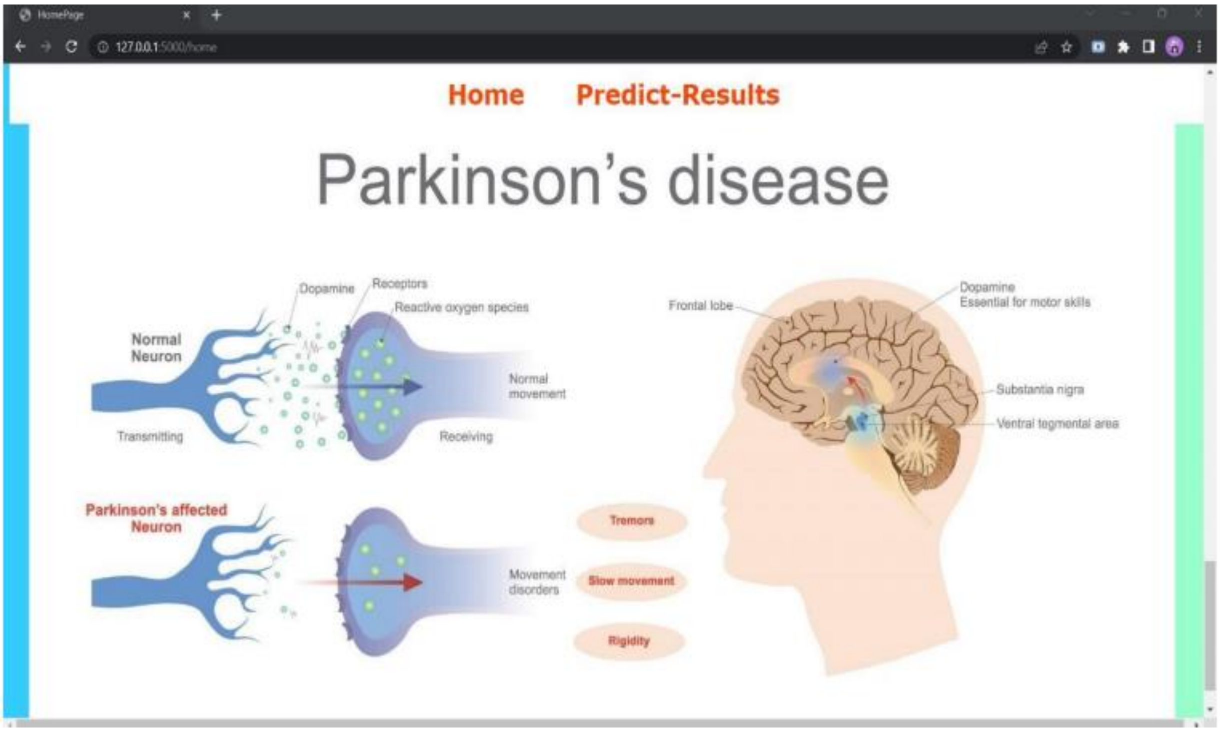
```python
            features = feature.hog(image, orientations=9,
                                   pixels_per_cell=(10, 10), cells_per_block=(2, 2),
                                   transform_sqrt=True, block_norm="L1")
            preds = modelw.predict([features])
            print(preds)
            ls = ["healthy", "parkinson"]
            result = ls[preds[0]]
            return result

        def pre_spiral(loc):
            file = os.listdir(loc)[0]
            img = os.path.join(loc, file)
            image = cv2.imread(img)
            output = image.copy()
            output = cv2.resize(output, (128, 128))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            image = cv2.resize(image, (200, 200))
            image = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

            features = feature.hog(image, orientations=9,
                                   pixels_per_cell=(10, 10), cells_per_block=(2, 2),
                                   transform_sqrt=True, block_norm="L1")
            preds = models.predict([features])
            print(preds)
            ls = ["healthy", "parkinson"]
            result = ls[preds[0]]
            return result

        locw = f'E:\parkinson_wave_spiral\Flask_App\Files\static\images\wave'
        locs = f'E:\parkinson_wave_spiral\Flask_App\Files\static\images\spiral'
        resultw = pre_wave(locw)
        results = pre_spiral(locs)

        return redirect(url_for('result_page', resultw=resultw, results=results))
    return render_template('index6.html', form=form)
```

OUTPUT

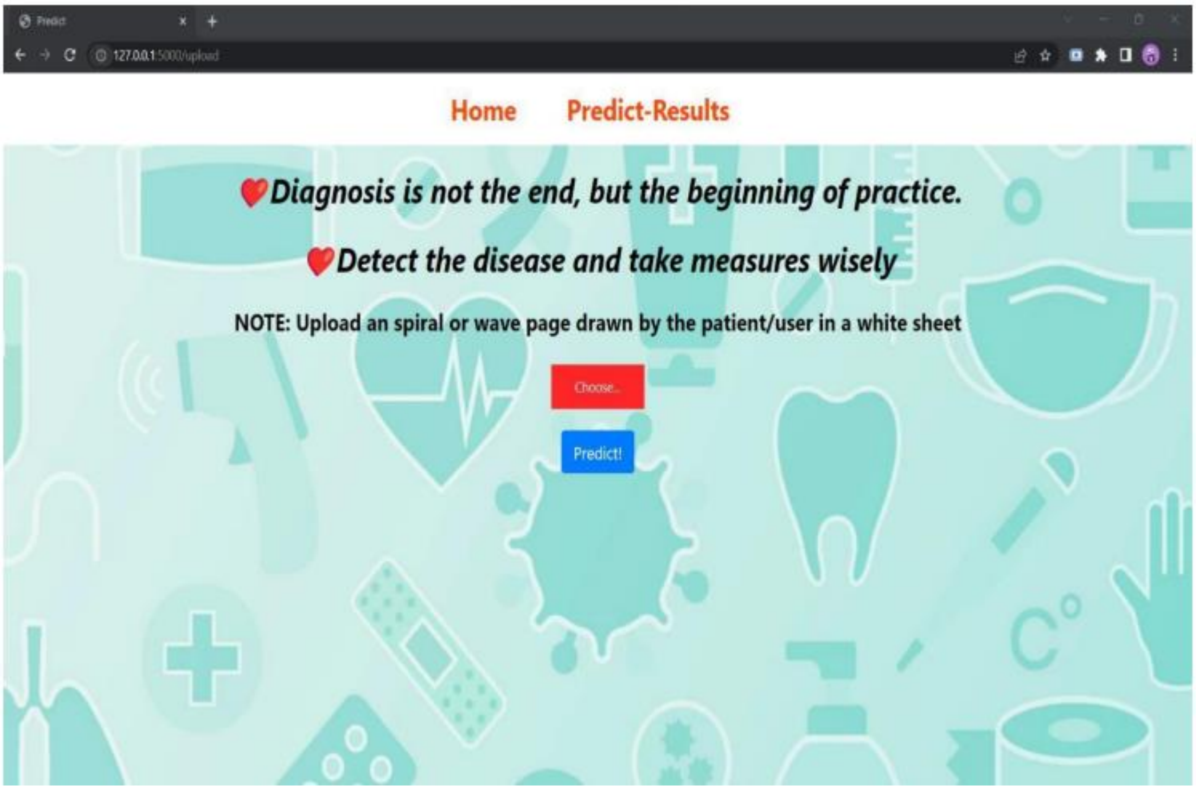
HOMEPAGE



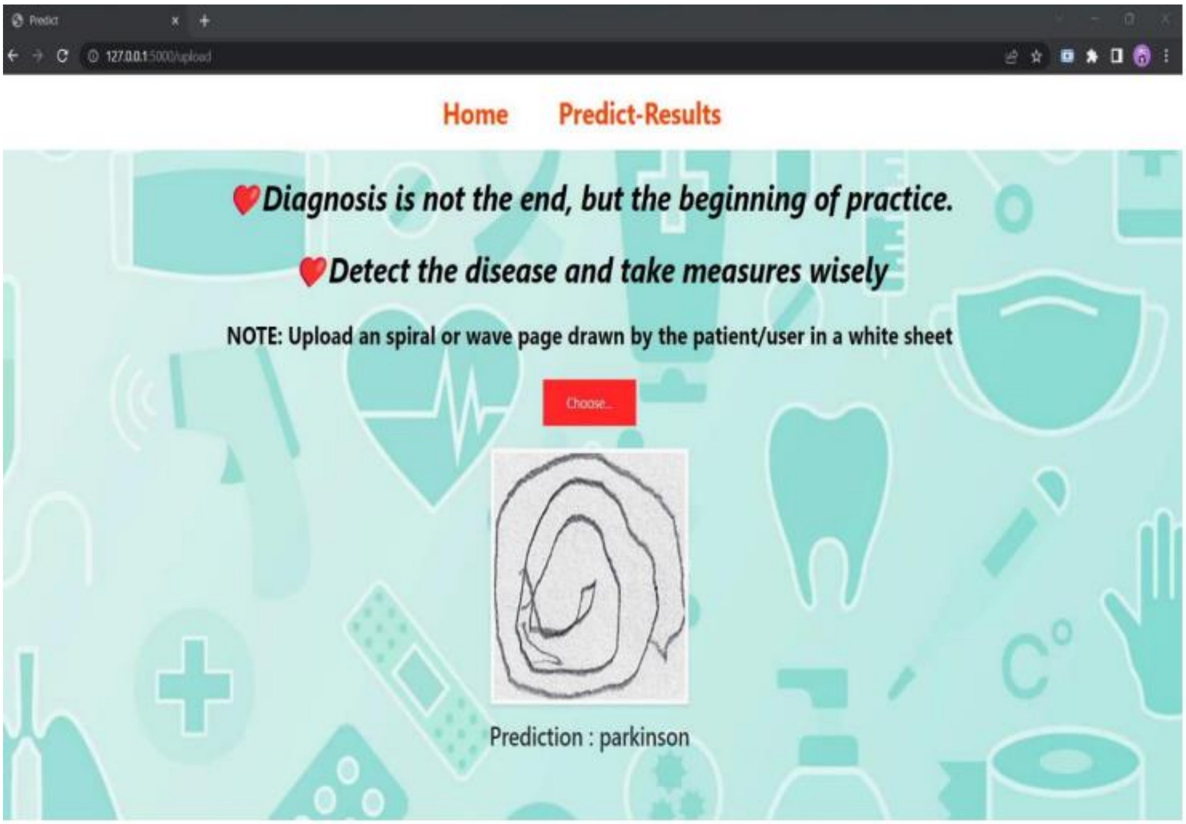

HomePage
127.0.0.1:5000/home
Home
Predict-Results
How brains looks during PD?
Healthy
Parkinson's Disease before symptoms
Parkinson's Disease after symptoms

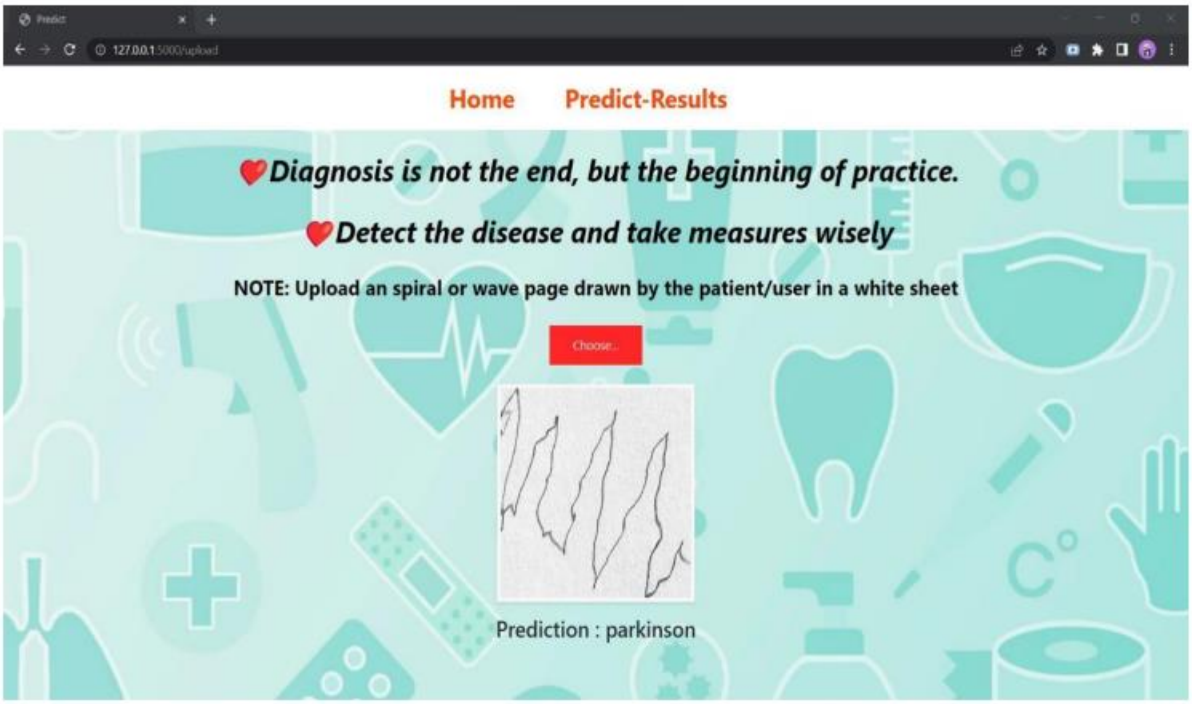

HomePage
127.0.0.1:5000/home
Home
Predict-Results
Parkinson's disease
Dopamine
Receptors
Reactive oxygen species
Normal Neuron
Normal movement
Transmitting
Receiving
Parkinson's affected Neuron
Movement disorders
Tremors
Slow movement
Rigidity
Frontal lobe
Dopamine Essential for motor skills
Substantia nigra
Ventral tegmental area

TEST VITAL PAGE



PREDICTED RESULT OF SPIRAL/WAVE IMAGES


Home
Predict-Results
Diagnosis is not the end, but the beginning of practice.
Detect the disease and take measures wisely
NOTE: Upload an spiral or wave page drawn by the patient/user in a white sheet
Prediction : parkinson


Home
Predict-Results
Diagnosis is not the end, but the beginning of practice.
Detect the disease and take measures wisely
NOTE: Upload an spiral or wave page drawn by the patient/user in a white sheet
Prediction : parkinson

CHAPTER 6

6. CONCLUSION

Parkinson Disorder is a disorder whose diagnosis is complex because of its symptoms similar to other disorders. Moreover, the lack of awareness increases the vulnerability of the patient's health. This often leads to misdiagnosis of the disorder. The diagnosis of Parkinson's Disease is not a straight-away-process which implies, a single test like ECG or blood test alone cannot determine PD in a person. Doctors need to study the patient's medical history followed by some neurological tests. With the high rate of misdiagnosis of PD, due to indefinite tests, leads to a crisis.Technology such as Data Science and Machine Learning tend to utilize this crisis as an opportunity to make diagnosis and treatment of PD patients easy.

.

REFERENCES

1.Gowthami Tallapureddy, D Radha, "Analysis of Ensemble of Machine Learning Algorithms for Detection of Parkinson's Disease", *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pp.354-361, 2022.

2. Pranita Kolhe, Kamlesh Kalbande, Atul Deshmukh, "Internet of Thing and Machine Learning Approach for Agricultural Application: A Review", *2022 10th International Conference on Emerging Trends in Engineering and Technology – Signal and Information Processing (ICETET-SIP-22)*, pp.1-6, 2022.

3. Kamal Nayan Reddy Challa, Venkata Sasank Pagolu, Ganapati Panda and Babita Majhi, *An Improved Approach for Prediction of Parkinson's Disease using Machine Learning Techniques*, Oct 2016, [online] Available

4. Shakya, Subarna, and Lalitpur Nepal. "Computational Enhancements of Wearable Healthcare Devices on Pervasive Computing System." Journal of Ubiquitous Computing and Communication Technologies (UCCT) 2, no. 02 (2020): 98-108.

4. An Improved Approach for Prediction of Parkinson’s Disease using Machine Learning Techniques, Kamal Nayan Reddy Challa, Venkata Sasank Pagolu, Ganapati Panda, Babita Majhi, arXiv:1610.08250v1 [cs.LG] 26 Oct 2016

5. Predication Of Parkinson’s Disease Using Data Mining Methods: A Comparative Analysis Of Tree, Statistical, And Support Vector Machine Classifiers [Article in Indian Journal of Medical Sciences · June 2011 DOI: 10.4103/0019-5359.107023 · Source: PubMed]

6. Collection and Analysis of a Parkinson Speech Dat aset With Multiple Types of Sound Recordings Betul Erdogdu Sakar, M. Erdem Isenkul, C. Okan Sakar, Ahmet Sertbas, Fikret Gurgen, Sakir Delil, Hulya Apaydin, and Olcay Kursun, Article in IEEE Journal of Biomedical and Health Informatics · July 2013

7. Predication of Parkinson's disease using data mining methods: A comparative analysis of tree, statistical and support vector machine classifiers, June 2011, Indian Journal of Medical Sciences 65(6):231-42, DOI: 10.4103/0019-5359.107023

8. Detecting Parkinson’s Disease with Machine Learning; medium.com/analytics-vidhya/detecting-parkinsons-disease-with-machine-learning-44c17208afce