# THE ASSAM KAZIRANGA UNIVERSITY

## MINOR PROJECT ON:

PARKINSON'S DISEASE DETECTION

**EXPERIMENT PAGES**

## GROUP MEMBERS:

| | | |
|---|---|---|
| 1. | PALLAVI NIMODIA | ET18BTHCS034 |
| 2. | SAIKAT KAMAL HALDER | ET18BTHCS022 |
| 3. | RITUSHMITA BORAH | ET18BTHEC004 |
| 4. | TANGRIK CH MARAK | ET18BTHEE006L |
| 5. | RECHAL L AYE | ET18BTHEE007L |

**TOOL(s) USED:** Python

# Description

A machine learning Based approach to detecting the presence of Parkison′s disease from spiral teats of patients. The dataset was obtained from Kaggle.

The dataset contains test of 195 people out of which 147 are suffering from disease and has 24 columns(name, mdvp:fo, mdvp:fhi, mdvp:flo, mdvp:jitter and so on). The data is in ASCII CSV format.

There are several classification algorithms such as Logistic Regression, Random Forest, etc. to approach the problem statement. For this project, we have used SVM approach as it gave the best results as we will see further.

# Importing the Libraries

The first and foremost step is to import all the important libraries and packages necessary for visualization and analysis.

In [127]: 8

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
X = StandardScaler()
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

# Data Collection cum Analysis

Second step is to upload the data and go through the dataset to identify any missing value and take necessary measures.

In [128]: 8

```
park_data=pd.read_csv('C:\parkinsons.csv')
park_data.head()
```

Out[128] :

| | name | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Ak |
|---|---|---|---|---|---|---|
| 0 | phon_R01_S01_1 | 119.992 | 157.302 | 74.997 | 0.00784 | 0.000 |
| 1 | phon_R01_S01_2 | 122.400 | 148.650 | 113.819 | 0.00968 | 0.000 |
| 2 | phon_R01_S01_3 | 116.682 | 131.111 | 111.555 | 0.01050 | 0.000 |
| 3 | phon_R01_S01_4 | 116.676 | 137.871 | 111.366 | 0.00997 | 0.000 |
| 4 | phon_R01_S01_5 | 116.014 | 141.781 | 110.655 | 0.01284 | 0.000 |

5 rows • 24 columns

In [129]: N

```
park_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype

 0   name              195 non-null    object
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
 15  NHR               195 non-null    float64
 16  HNR               195 non-null    float64
 17  status            195 non-null    int64
 18  RPDE              195 non-null    float64
 19  DFA               195 non-null    float64
 20  spread1           195 non-null    float64
 21  spread2           195 non-null    float64
 22  D2                195 non-null    float64
 23  PPE               195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

In [130]: 8

```
park_data.shape
```

Out[130]:

```
(195, 24)
```

In [131]: |H

```
park_data.isnull().sum()
```

Out[131]:

```
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
spreadl             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

In [132]: |H

```
park_data . descr1be ( )
#Here ue get the stottstcs o/ the  data
```

Out[132]:

| | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RA |
|---|---|---|---|---|---|---|
| count | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.00000 |
| mean | 154.228641 | 197.104918 | 116.324631 | 0.006220 | 0.000044 | 0.00330 |
| std | 41.390065 | 91.491548 | 43.521413 | 0.004848 | 0.000035 | 0.00296 |
| min | 88.333000 | 102.145000 | 65.476000 | 0.001680 | 0.000007 | 0.00068 |
| 25% | 117.572000 | 134.862500 | 84.291000 | 0.003460 | 0.000020 | 0.00166 |
| 50% | 148.790000 | 175.829000 | 104.315000 | 0.004940 | 0.000030 | 0.00250 |
| 75% | 182.769000 | 224.205500 | 140.018500 | 0.007365 | 0.000060 | 0.00383 |
| max | 260.105000 | 592.030000 | 239.170000 | 0.033160 | 0.000260 | 0.02144 |

8 rows 23 columns

In [133] : N

```
#target vans ab Le counts
park_data['status'].value_counts()
```

Out [133]:

```
1     147
0      48
Name: status, dtype: int64
```

# 1--> Parkinson's Disease Affected

# 0--> No Parkinson's Disease

In [134]: 8

```
park_data.groupby('status').mean()
```

Out[134]:

| | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RR |
|---|---|---|---|---|---|---|
| status | | | | | | |
| 0 | 181.937771 | 223.636750 | 145.207292 | 0.003866 | 0.000023 | 0.00191 |
| 1 | 145.180762 | 188.441463 | 106.893558 | 0.006989 | 0.000051 | 0.0037! |

2 rows    22 columns

# Preprocessing Stage

## Feature and Target separation

In [135]:

```
A = park_data.drop(columns=['name','status'],axis=1)
B = park_data['status']
```

In [136]:

```
print(A)
```

```
     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0        119.992       157.302        74.997         0.00784
1        122.400       148.650       113.819         0.00968
2        116.682       131.111       111.555         0.01050
3        116.676       137.871       111.366         0.00997
4        116.014       141.781       110.655         0.01284

190      174.188       230.978        94.261         0.00459
191      209.516       253. 017       89. 488        0.00564
192      174. 688      240.005        74.287         0.01360
193      198.764       396.961        74.904         0.00740
194      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

     MDVP:Shimmer(dB)      MDVP:APQ  Shimmer:DDA      NHR      HNR
RPDE  \
0               8. 426        0.02971      0.06545  0.02211  21.033  0.41
4783
1               0.626        0.04368      0.09403  0.01929  19.085  0.45
8359
2               0.482        0.03590      0.08270  0.01309  20.651  0.42
9895
3               0.517        0.03772      0.08771  0.01353  20.644  0.43
4969
4               0.584        0.04465      0.10470  0.01767  19.649  0.41
7356


190             0.405        0.02745      0.07008  0.02764  19.517  0.44
8439
191             0.263        0.01879      0.04812  0.01810  19.147  0.43
1674
192             0.256        0.01667      0.03804  0.10715  17.883  0.40
7567
193             0.241        0.01588      0.03794  0.07223  19.020  0.45
1221
194             0.190        0.01373      0.03078  0.04398  21.209  0.46
2803

          DFA   spreadl   spread2        D2       PPE
0    0.815285 -4.813031  0.266482  2.301442  0.284654
1    0.819521 -4.075192  0.335590  2.486855  0.368674
```

```
2    0.825288 -4.443179 0.311173 2.342259 0.332634
3    0.819235 -4.117501 0.334147 2.405554 0.368975
4    0.823484 -3.747787 0.234513 2.332180 0.410335

190  0.657899 -6.538586 0.121952 2.657476 0.133050
191  0.683244 -6.195325 0.129303 2.784312 0.168895
192  0.655683 -6.787197 0.158453 2.679772 0.131728
193  0.643956 -6.744577 0.207454 2.138608 0.123306
194  0.664357 -5.724056 0.190667 2.555477 0.148569

[195 rows x 22 columns]
```

In [137]: N

```
print(B)
```

```
0      1
1      1
2      1
3      1
4      1

190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```

# Splitting of Test and Training Datasets

Splitting the dataset into training and testing sets keeping 20% of the data for testing.

In [138]: N

```
X_train,X_test,Y_train,Y_test=train_test_split(A,B,test_size=0.2,random_sta
```

In [139]: 8

```
print(A.shape,X_train.shape,X_test.shape)
```

```
(195, 22) (156, 22) (39, 22)
```

**We took test size 20 percentage of the total dataset**

# Standardization of Data

In [140]: §

```
sc = StandardScaler()
```

In [141]: H

```
sc.fit(X_train)
```

Out[141]:

```
StandardScaler()
```

In [142]: fl

```
X_train = sc.transform(X_train)
X_test  = sc.transform(X_test)
```

In [143]: N

```
print(X_train)
```

```
[[ 0.63239631 -0.02731081 -0.87985049 ... -0.97586547 -0.55160318
   0.07769494]
 [-1.05512719 -0.83337041 -0.9284778  ...  0.3981808  -0.61014073
   0.39291782]
 [ 0.02996187 -0.29531068 -1.12211107 ... -0.43937044 -0.62849605
  -0.50948408]

 [-0.9096785  -0.6637302  -0.160638   ...  1.22001022 -0.47404629
  -0.2159482 ]
 [-0.35977689  0.19731822 -0.79063679 ... -0.17896029 -0.47272835
   0.28181221]
 [ 1.01957066  0.19922317 -0.61914972 ... -0.716232    1.23632066
  -0.05829386]]
```

# Model Training

## SVM Model

In [144]: N

```
model = svm.SVC(kernel='linear')
```

Classifier has Classes and Regression may have **integer or floating values**

In [145]: $

```
model.fit(X_train,Y_train)
#Here we fee   the  train ing dataset to the mode L
```

Out[145]:

```
SVC(kernel='linear')
```

# Model Evaluation

## Accuracy of the Model

In [146]: g

```
# accuracy oy the training data
X_train_pred = model.predict(X_train)
training_data_accuracy    accuracy_score(Y_train, X_train_pred)
```

In [147]: N

```
print('Training Data Accuracy :',training_data_accuracy)
```

```
Training Data Accuracy : 0.8846153846153846
```

In [148]: N

```
# accuracy oy ‹he test data
X_test_pred = model.predict(X_test)
test_data_accuracy    accuracy_score(Y_test, X_test_pred)
```

In [149]: N

```
print('Test Data Accuracy :',test_data_accuracy)
```

```
Test Data Accuracy : 0.8717948717948718
```

The output model of SVM shows 87% accuracy for the given dataset.

# Building the Prediction System

Now, we built the prediction model on the basis of the analysis done.

In [150]: N

```
input_data = (116.68200,131.11100,111.55500,0.01050,0.00009,0.00544,0.00781,£
#convens i on to numpy  array
input_data_as_numpy_array = np.asarray(input_data)
#reshape  for   it   needs on Ly one set of vaL ues
input_data_reshaped    input_data_as_numpy_array.reshape(1,-1)
#standardi z i ng

std_data = X.fit_transform(input_data_reshaped)
prediction = model.predict(std_data)
if (prediction[0] ==0):
    print("Not Parkinson's Disease Affected“)
else:
    print(“Parkinson's Disease Affected")
print(prediction)
```

```
Parkinson's Disease Affected
[1]
```

In [ ]: b

```
```

In [ ]: |4

```
```

In [ ]: b

```
```

# Conclusion

In this project, SVM classification model is used to predict the status of the patient. The accuracy for training set is 88% and testing set is 87%.