#Parkinson's Disease Detection

Group Number: 13

Students:

1. S.A Wijesinghe

   EG/2021/4877

---

1. I.U Madakaladeniya

   EG/2021/4651

Introduction

Parkinson's disease (PD) is the second most prevalent age-related neurological condition which causes a variety of physical and mental symptoms. It is challenging to diagnose Parkinson's disease (PD) because its symptoms are quite similar to those of other illnesses, like essential tremor and normal ageing. When a person reaches the age of 50, outward signs like trouble walking and speaking start to appear. Certain drugs relieve some of the symptoms of Parkinson's disease, despite the fact that there is a treatment available. By managing the complications brought on by the disease, patients are able to live their normal lives. Identifying the disease and stopping its progression are crucial at this stage.Parkinson's disease (PD) is a steadily deteriorating condition with symptoms that gradually arise over time, affects millions of people globally. About 10% of people exhibit symptoms of the disease before the age of 40, whereas prominent symptoms arise for those above 50 1.

This project aims to build machine learning models to detect Parkinson's disease using biomedical voice measurements. In order to distinguish between healthy and PD patients based on voice signal characteristics, our project uses a two Machine Learning (ML) models, including Support Vector Machine (SVM), Random Forest Classifier(RF),195 voice recordings of examinations performed on 31 patients made up the dataset. In order to improve model performance, our models was trained using feature selection, hyperparameter tuning (GridSearchCV), feature scaling, and the Random Over-sampler.Evaluation techniques and metrics, including the Classification Report, F1-Score, Accuracy, Precision, Recall, and Confusion Matrix were used to evaluate the model performance.

Literature Survey

Parkinson's Disease is named after James Parkinson a British physician in 1817 3. It has been proven that machine learning techniques are effective in detecting Parkinson's disease early on. These algorithms have been used for many years to diagnose diseases. For example, Lafunte and his colleagues used ANN in 1997 to separate individuals with lower limb arthritis from those in good health with 80% accuracy 4. Using a collection of biological voice signals from both healthy and Parkinson samples, A. Sharma et al. (2014) applied pattern recognition, neural networks, and support vector machines to diagnose Parkinson's disease (PD) with an accuracy of 85.294% 5. Another study was conducted in 2020, and the results showed that PD patients could be differentiated from healthy people at an early stage of the disease with accuracy rates of 93.5%, 96%, and 95.2% using features such as RMS, chroma STFT, spectral centroid, etc 6.

Dataset Description

This dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each column in the table is a particular voice measure, and each row corresponds to one of 195 voice recordings from these individuals ("name" column). The main aim of the data is to discriminate healthy people from those with PD, according to the "status" column which is set to 0 for healthy and 1 for PD.

Matrix column entries (attributes):

- name - ASCII subject name and recording number
- MDVP:Fo(Hz) - Average vocal fundamental frequency
- MDVP:Fhi(Hz) - Maximum vocal fundamental frequency
- MDVP:Flo(Hz) - Minimum vocal fundamental frequency
- MDVP:Jitter(%),MDVP:Jitter(Abs),MDVP:RAP,MDVP:PPQ,Jitter:DDP - Several measures of variation in fundamental frequency
- MDVP:Shimmer,MDVP:Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,MDVP:APQ,Shimmer:DDA - Several measures of variation in amplitude
- NHR,HNR - Two measures of ratio of noise to tonal components in the voice
- status - Health status of the subject (one) - Parkinson's, (zero) - healthy
- RPDE,D2 - Two nonlinear dynamical complexity measures
- DFA - Signal fractal scaling exponent
- spread1,spread2,PPE - Three nonlinear measures of fundamental frequency variation'

Exploratory Data Analysis

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Importing the libraries

```
import os,sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
sns.set()
import warnings
warnings.filterwarnings('ignore')
```

```
!pip install plotly
```

```
Requirement already satisfied: plotly in
/usr/local/lib/python3.10/dist-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly) (9.0.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from plotly) (24.2)
```

```
#to check the current path
os.getcwd()
```

{"type":"string"}

```
pd.set_option('display.max_rows',200)
pd.set_option('display.max_column',25)
pd.set_option('display.width',200)
```

```
# Load Data
data_path = "/content/drive/MyDrive/ML Project/parkinsons data.csv"
df = pd.read_csv(data_path)
print("Dataset Loaded. First 5 rows:")
print(df.head())
```

```
Dataset Loaded. First 5 rows:
             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  \
0  phon_R01_S01_1      119.992       157.302        74.997
0.00784           0.00007   0.00370   0.00554     0.01109
0.04374             0.426       0.02182       0.03130   0.02971
1  phon_R01_S01_2      122.400       148.650       113.819
0.00968           0.00008   0.00465   0.00696     0.01394
0.06134             0.626       0.03134       0.04518   0.04368
2  phon_R01_S01_3      116.682       131.111       111.555
0.01050           0.00009   0.00544   0.00781     0.01633
0.05233             0.482       0.02757       0.03858   0.03590
3  phon_R01_S01_4      116.676       137.871       111.366
0.00997           0.00009   0.00502   0.00698     0.01505
0.05492             0.517       0.02924       0.04005   0.03772
4  phon_R01_S01_5      116.014       141.781       110.655
0.01284           0.00011   0.00655   0.00908     0.01966
0.06425             0.584       0.03490       0.04825   0.04465

   Shimmer:DDA      NHR      HNR  status      RPDE       DFA   spread1
spread2        D2       PPE
```

```
0      0.06545  0.02211  21.033       1  0.414783  0.815285 -4.813031
0.266482  2.301442  0.284654
1      0.09403  0.01929  19.085       1  0.458359  0.819521 -4.075192
0.335590  2.486855  0.368674
2      0.08270  0.01309  20.651       1  0.429895  0.825288 -4.443179
0.311173  2.342259  0.332634
3      0.08771  0.01353  20.644       1  0.434969  0.819235 -4.117501
0.334147  2.405554  0.368975
4      0.10470  0.01767  19.649       1  0.417356  0.823484 -3.747787
0.234513  2.332180  0.410335
```

Data Preprocessing

Find NULL values

```
print("\nMissing values in the dataset:")
print(df.isnull().sum())
```

```
Missing values in the dataset:
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
```

There are no missing values in the dataset so, we don't need the step to impute missing values.

Checking the data types of the columns

```
# Data types of columns
print("\nData types of columns:")
print(df.dtypes)
```

```
Data types of columns:
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
DFA                  float64
spread1              float64
spread2              float64
D2                   float64
PPE                  float64
dtype: object
```

All the variables are numerical and there are no categorical variables, so we don't need to handle categorical variables.

Checking whether there are any unnessesary characters in the data set.

```
for i in df.columns:

print("**********************************************",i,"************
***********************************")
  print()
  print(set(df[i].tolist()))
  print()
```

```
********************************************** name
***********************************************
```

{'phon_R01_S43_3', 'phon_R01_S39_5', 'phon_R01_S10_3',
'phon_R01_S27_4', 'phon_R01_S04_5', 'phon_R01_S02_5',
'phon_R01_S26_5', 'phon_R01_S33_4', 'phon_R01_S07_2',
'phon_R01_S17_3', 'phon_R01_S20_1', 'phon_R01_S05_3',
'phon_R01_S34_3', 'phon_R01_S37_2', 'phon_R01_S08_4',
'phon_R01_S34_1', 'phon_R01_S34_2', 'phon_R01_S44_3',
'phon_R01_S16_3', 'phon_R01_S07_1', 'phon_R01_S16_6',
'phon_R01_S32_5', 'phon_R01_S10_6', 'phon_R01_S07_6',
'phon_R01_S27_3', 'phon_R01_S22_1', 'phon_R01_S07_5',
'phon_R01_S37_1', 'phon_R01_S06_5', 'phon_R01_S25_3',
'phon_R01_S32_4', 'phon_R01_S27_7', 'phon_R01_S35_1',
'phon_R01_S44_1', 'phon_R01_S25_5', 'phon_R01_S31_2',
'phon_R01_S42_4', 'phon_R01_S50_6', 'phon_R01_S42_5',
'phon_R01_S18_1', 'phon_R01_S01_4', 'phon_R01_S20_2',
'phon_R01_S31_1', 'phon_R01_S18_6', 'phon_R01_S18_3',
'phon_R01_S43_1', 'phon_R01_S21_3', 'phon_R01_S08_2',
'phon_R01_S18_5', 'phon_R01_S05_1', 'phon_R01_S07_3',
'phon_R01_S01_5', 'phon_R01_S18_2', 'phon_R01_S39_3',
'phon_R01_S37_6', 'phon_R01_S05_4', 'phon_R01_S27_1',
'phon_R01_S13_5', 'phon_R01_S39_6', 'phon_R01_S43_2',
'phon_R01_S17_1', 'phon_R01_S25_6', 'phon_R01_S02_3',
'phon_R01_S35_6', 'phon_R01_S33_3', 'phon_R01_S24_3',
'phon_R01_S05_6', 'phon_R01_S01_3', 'phon_R01_S06_4',
'phon_R01_S17_6', 'phon_R01_S33_1', 'phon_R01_S42_1',
'phon_R01_S02_2', 'phon_R01_S32_2', 'phon_R01_S32_3',
'phon_R01_S19_5', 'phon_R01_S13_4', 'phon_R01_S10_1',
'phon_R01_S16_2', 'phon_R01_S24_1', 'phon_R01_S37_4',
'phon_R01_S05_5', 'phon_R01_S17_2', 'phon_R01_S16_1',
'phon_R01_S27_6', 'phon_R01_S19_4', 'phon_R01_S35_2',
'phon_R01_S07_4', 'phon_R01_S50_2', 'phon_R01_S04_6',
'phon_R01_S22_3', 'phon_R01_S49_2', 'phon_R01_S25_1',
'phon_R01_S26_1', 'phon_R01_S49_6', 'phon_R01_S35_7',
'phon_R01_S33_2', 'phon_R01_S02_1', 'phon_R01_S37_3',
'phon_R01_S08_6', 'phon_R01_S04_3', 'phon_R01_S22_5',
'phon_R01_S16_5', 'phon_R01_S21_6', 'phon_R01_S05_2',
'phon_R01_S08_5', 'phon_R01_S32_1', 'phon_R01_S19_6',
'phon_R01_S04_1', 'phon_R01_S06_1', 'phon_R01_S19_1',
'phon_R01_S35_5', 'phon_R01_S08_1', 'phon_R01_S44_2',
'phon_R01_S25_2', 'phon_R01_S34_5', 'phon_R01_S44_4',
'phon_R01_S06_3', 'phon_R01_S44_6', 'phon_R01_S16_4',
'phon_R01_S22_6', 'phon_R01_S39_2', 'phon_R01_S26_3',
'phon_R01_S33_5', 'phon_R01_S19_2', 'phon_R01_S10_5',
'phon_R01_S33_6', 'phon_R01_S35_4', 'phon_R01_S49_3',
'phon_R01_S17_5', 'phon_R01_S21_5', 'phon_R01_S08_3',
'phon_R01_S26_6', 'phon_R01_S39_1', 'phon_R01_S01_2',
'phon_R01_S19_3', 'phon_R01_S42_6', 'phon_R01_S21_7',
'phon_R01_S50_3', 'phon_R01_S22_4', 'phon_R01_S43_5',
'phon_R01_S49_5', 'phon_R01_S34_4', 'phon_R01_S32_6',
'phon_R01_S24_5', 'phon_R01_S10_2', 'phon_R01_S20_6',

```
'phon_R01_S06_6', 'phon_R01_S35_3', 'phon_R01_S24_2',
'phon_R01_S43_4', 'phon_R01_S06_2', 'phon_R01_S31_6',
'phon_R01_S49_4', 'phon_R01_S42_3', 'phon_R01_S21_2',
'phon_R01_S01_6', 'phon_R01_S13_6', 'phon_R01_S04_2',
'phon_R01_S25_4', 'phon_R01_S50_5', 'phon_R01_S31_5',
'phon_R01_S31_4', 'phon_R01_S13_1', 'phon_R01_S13_2',
'phon_R01_S20_4', 'phon_R01_S43_6', 'phon_R01_S01_1',
'phon_R01_S50_1', 'phon_R01_S20_5', 'phon_R01_S27_5',
'phon_R01_S24_4', 'phon_R01_S10_4', 'phon_R01_S21_1',
'phon_R01_S20_3', 'phon_R01_S02_4', 'phon_R01_S17_4',
'phon_R01_S22_2', 'phon_R01_S34_6', 'phon_R01_S49_1',
'phon_R01_S24_6', 'phon_R01_S26_2', 'phon_R01_S04_4',
'phon_R01_S44_5', 'phon_R01_S27_2', 'phon_R01_S13_3',
'phon_R01_S39_4', 'phon_R01_S37_5', 'phon_R01_S18_4',
'phon_R01_S21_4', 'phon_R01_S26_4', 'phon_R01_S50_4',
'phon_R01_S42_2', 'phon_R01_S02_6', 'phon_R01_S31_3'}

********************************************** 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, 108.664, 123.109, 110.019, 123.925, 112.24, 113.84,
112.777, 115.871, 115.697, 125.306, 125.213, 124.393, 120.103,
113.597, 122.611, 126.632, 123.723, 125.394, 126.358, 127.533,
128.611, 129.916, 130.049, 131.111, 131.162, 132.068, 134.231,
135.069, 134.656, 137.871, 137.244, 138.052, 139.867, 141.781, 139.71,
143.946, 140.557, 141.756, 141.068, 130.27, 148.65, 131.067, 131.897,
148.826, 150.449, 133.344, 154.609, 131.731, 128.442, 157.302,
157.765, 159.866, 159.774, 161.469, 162.215, 163.305, 162.824,
165.738, 166.607, 162.408, 163.335, 164.989, 163.267, 168.913, 172.86,
172.975, 135.738, 175.829, 176.595, 134.209, 177.291, 179.139,
129.038, 138.752, 142.369, 185.604, 142.83, 139.644, 189.398, 190.204,
191.759, 192.735, 193.221, 192.921, 195.107, 196.537, 197.724,
198.346, 197.173, 200.841, 201.249, 202.324, 200.125, 202.45, 205.56,
206.002, 206.896, 208.313, 208.701, 209.512, 211.604, 211.526,
210.565, 211.35, 215.203, 208.9, 217.455, 215.293, 219.29, 220.315,
221.3, 216.814, 223.982, 216.302, 225.93, 226.355, 227.383, 227.381,
224.429, 230.978, 231.345, 232.181, 232.706, 234.619, 233.481,
231.508, 237.494, 238.987, 239.541, 233.099, 241.35, 240.005, 243.709,
244.663, 245.135, 247.326, 248.834, 250.912, 252.221, 253.792,
253.017, 255.034, 126.609, 260.277, 261.487, 262.09, 263.872, 154.284,
264.919, 262.707, 268.796, 155.982, 271.314, 272.21, 157.339, 158.359,
127.349, 160.267, 160.368, 144.466, 161.078, 163.736, 163.441,
163.417, 133.374, 349.259, 396.961, 128.101, 127.611, 442.557,
442.824, 450.247, 479.697, 197.238, 198.109, 198.966, 492.892,
203.522}

********************************************** MDVP:Flo(Hz)
**********************************************

{102.137, 104.437, 104.773, 104.095, 105.667, 105.554, 105.715,
106.821, 106.656, 107.816, 107.802, 108.634, 108.97, 109.216, 109.836,
109.815, 112.773, 112.173, 113.201, 65.75, 65.809, 65.782, 68.623,
67.021, 66.004, 67.343, 65.476, 68.401, 74.997, 75.836, 76.556, 77.63,
75.603, 78.128, 79.068, 76.779, 77.968, 82.764, 83.159, 84.072,
81.737, 86.292, 86.18, 80.055, 90.264, 91.226, 91.754, 85.545, 87.549,
95.628, 87.804, 96.206, 98.664, 99.77, 92.02, 93.978, 102.874, 103.37,
104.68, 105.007, 106.981, 107.024, 107.316, 108.153, 109.379, 110.402,
111.208, 104.315, 110.655, 111.366, 111.555, 113.819, 113.787, 114.82,
115.765, 114.676, 122.08, 117.495, 118.604, 125.61, 128.621, 129.859,
131.276, 132.857, 133.608, 133.751, 135.041, 138.99, 141.047, 142.822,
142.299, 144.878, 144.811, 144.148, 147.226, 148.691, 149.605,
149.442, 151.451, 144.736, 144.786, 155.495, 161.34, 163.564, 164.168,
165.982, 166.977, 168.013, 168.793, 173.015, 174.478, 175.456,
177.584, 177.258, 182.786, 185.258, 100.139, 189.621, 192.055,
192.091, 193.104, 195.708, 196.16, 197.079, 141.998, 199.02, 205.495,
219.783, 221.156, 223.634, 225.227, 227.911, 229.256, 231.848,
232.483, 232.435, 237.303, 239.17, 66.157, 100.757, 69.085, 71.948,
```

```
116.346, 74.677, 74.287, 74.904, 75.501, 75.344, 75.632, 75.349,
76.596, 77.022, 77.973, 78.032, 78.228, 79.032, 79.187, 79.82, 79.512,
79.543, 80.297, 80.637, 81.114, 82.063, 83.961, 83.34, 84.51, 85.902,
86.795, 86.232, 86.228, 86.647, 87.638, 88.251, 88.833, 89.686,
89.488, 90.794, 91.802, 91.121, 93.105, 100.673, 94.794, 94.246,
94.261, 95.654, 96.913, 96.983, 97.543, 97.527, 98.25, 99.923, 99.503,
116.187, 100.209}

*********************************************** MDVP:Jitter(%)
***********************************************

{0.00766, 0.00505, 0.00183, 0.00349, 0.00532, 0.0021, 0.00332,
0.00254, 0.00376, 0.00298, 0.00742, 0.00803, 0.00281, 0.00342,
0.01813, 0.00264, 0.00647, 0.00752, 0.00369, 0.00874, 0.00352,
0.00718, 0.0084, 0.01101, 0.00396, 0.01284, 0.00257, 0.00762, 0.0044,
0.00684, 0.00867, 0.00606, 0.00284, 0.0105, 0.00406, 0.00589, 0.00267,
0.00633, 0.00694, 0.00494, 0.00555, 0.00294, 0.00355, 0.02714,
0.00277, 0.0046, 0.00321, 0.00382, 0.00704, 0.03107, 0.00609, 0.00531,
0.00975, 0.0027, 0.00331, 0.00392, 0.0128, 0.00314, 0.00436, 0.00619,
0.00358, 0.01568, 0.0048, 0.00419, 0.01551, 0.00968, 0.00524, 0.00907,
0.00768, 0.00185, 0.0049, 0.00551, 0.00168, 0.0029, 0.01378, 0.00856,
0.00534, 0.00212, 0.00517, 0.01466, 0.00761, 0.005, 0.00178, 0.00544,
0.00605, 0.00788, 0.00405, 0.00727, 0.00971, 0.00205, 0.00266,
0.00327, 0.00571, 0.0031, 0.00432, 0.00293, 0.00476, 0.0052, 0.00459,
0.00198, 0.00581, 0.00842, 0.00381, 0.00442, 0.00564, 0.00303,
0.00747, 0.00225, 0.00608, 0.03011, 0.03316, 0.00757, 0.00496,
0.00174, 0.00235, 0.00296, 0.0074, 0.0054, 0.00923, 0.00784, 0.01872,
0.00462, 0.00567, 0.00428, 0.00289, 0.00733, 0.00411, 0.00533,
0.00472, 0.0136, 0.01038, 0.00333, 0.00455, 0.00516, 0.00638, 0.00316,
0.00238, 0.00621, 0.01936, 0.00282, 0.00404, 0.00709, 0.00448,
0.00831, 0.01719, 0.00248, 0.00692, 0.00309, 0.00431, 0.0037, 0.00492,
0.00997, 0.00353, 0.00397, 0.00336, 0.00841, 0.00519, 0.00258,
0.00702, 0.00502, 0.0018, 0.00241, 0.00346, 0.00407, 0.00651, 0.0039,
0.00451, 0.00817, 0.00495, 0.00356, 0.01627, 0.00417, 0.00339}

*********************************************** MDVP:Jitter(Abs)
***********************************************

{0.00011, 0.00022, 5e-05, 0.00016, 0.0001, 4e-05, 0.00015, 0.00026,
9e-06, 7e-06, 9e-05, 3e-05, 0.00014, 8e-05, 2e-05, 7e-05, 1e-05,
0.00012, 6e-05}

*********************************************** MDVP:RAP
***********************************************

{0.01854, 0.00244, 0.00366, 0.00105, 0.00166, 0.00349, 0.00593,
0.01159, 0.00393, 0.00254, 0.00115, 0.00176, 0.00237, 0.00159, 0.0022,
0.00281, 0.00403, 0.00647, 0.00186, 0.00247, 0.0043, 0.00169, 0.00996,
0.00291, 0.00352, 0.00152, 0.00135, 0.00196, 0.00257, 0.00118,
0.00284, 0.00467, 0.00406, 0.00206, 0.00389, 0.0045, 0.00189, 0.0025,
```

```
0.00372, 0.00233, 0.00294, 0.00094, 0.00155, 0.00277, 0.0026, 0.00321,
0.00826, 0.00121, 0.00182, 0.00165, 0.00226, 0.00287, 0.00209, 0.0027,
0.00331, 0.00131, 0.00114, 0.00175, 0.01568, 0.00863, 0.00219, 0.0028,
0.00463, 0.00141, 0.00202, 0.00263, 0.00124, 0.00507, 0.00368, 0.0049,
0.00168, 0.02144, 0.01117, 0.00351, 0.00412, 0.00534, 0.00334,
0.00134, 0.00117, 0.00178, 0.00622, 0.001, 0.00544, 0.00849, 0.00144,
0.00205, 0.00388, 0.00127, 0.00371, 0.00493, 0.00171, 0.00232,
0.00293, 0.00415, 0.00093, 0.00154, 0.00398, 0.00076, 0.00137,
0.00181, 0.00364, 0.00469, 0.00147, 0.00269, 0.00391, 0.00191,
0.00113, 0.00174, 0.00418, 0.00157, 0.00279, 0.00201, 0.00506,
0.00428, 0.00211, 0.00655, 0.00316, 0.00116, 0.00238, 0.00299,
0.00743, 0.0016, 0.00221, 0.00465, 0.00404, 0.00204, 0.00387, 0.01075,
0.0037, 0.00109, 0.0017, 0.00414, 0.00092, 0.00153, 0.00919, 0.00214,
0.00075, 0.00136, 0.0038, 0.00502, 0.0018, 0.00241, 0.00624, 0.00302,
0.00163, 0.00224, 0.018, 0.00146, 0.00268, 0.00068, 0.00373, 0.00173,
0.00295, 0.00356, 0.00905}

********************************************* MDVP:PPQ
*********************************************

{0.00122, 0.00461, 0.00183, 0.00244, 0.00166, 0.00227, 0.00149,
0.00332, 0.00454, 0.00576, 0.00254, 0.00698, 0.00315, 0.00115,
0.00176, 0.00237, 0.0042, 0.00159, 0.0022, 0.00908, 0.00203, 0.00186,
0.00247, 0.00169, 0.00152, 0.00213, 0.00718, 0.00396, 0.00135,
0.00196, 0.00318, 0.0044, 0.00623, 0.00162, 0.00284, 0.00467, 0.00267,
0.00389, 0.0045, 0.00128, 0.00233, 0.00155, 0.00399, 0.00138, 0.00199,
0.00182, 0.01958, 0.00226, 0.00348, 0.00148, 0.0027, 0.00514, 0.00453,
0.00192, 0.00253, 0.00314, 0.00375, 0.00819, 0.00175, 0.00419, 0.0028,
0.00463, 0.00202, 0.00263, 0.00385, 0.00246, 0.00107, 0.00168, 0.0029,
0.00351, 0.00151, 0.00395, 0.00134, 0.00256, 0.00317, 0.003, 0.00422,
0.001, 0.00283, 0.00144, 0.00205, 0.00327, 0.01154, 0.00449, 0.00188,
0.00371, 0.00432, 0.00554, 0.00493, 0.00171, 0.00232, 0.00354,
0.00415, 0.00215, 0.00337, 0.00781, 0.00398, 0.00137, 0.0052, 0.00198,
0.00259, 0.00564, 0.00486, 0.002, 0.00469, 0.00147, 0.00208, 0.00269,
0.0033, 0.00696, 0.00113, 0.00235, 0.01628, 0.00096, 0.00218, 0.0034,
0.0014, 0.00184, 0.0075, 0.00428, 0.00106, 0.00167, 0.00289, 0.01699,
0.00211, 0.00655, 0.00133, 0.00194, 0.00316, 0.00238, 0.00221,
0.00909, 0.00387, 0.00448, 0.0017, 0.00231, 0.00292, 0.00092, 0.00153,
0.00275, 0.00336, 0.00136, 0.00197, 0.00258, 0.00963, 0.00319,
0.00946, 0.00241, 0.00485, 0.0099, 0.00346, 0.00207, 0.00329, 0.0039,
0.0019, 0.01522, 0.00312, 0.00173, 0.00234, 0.00539, 0.00478, 0.00339,
0.00139, 0.01027, 0.00261}

********************************************* Jitter:DDP
*********************************************

{0.06433, 0.00949, 0.00749, 0.01193, 0.0081, 0.00488, 0.01254,
0.00671, 0.00349, 0.00471, 0.00715, 0.00393, 0.00898, 0.00837,
0.00315, 0.00498, 0.01003, 0.00742, 0.00803, 0.00542, 0.00342,
0.00403, 0.00508, 0.02589, 0.01057, 0.01179, 0.00535, 0.00457,
```

```
0.00962, 0.00762, 0.00301, 0.01067, 0.01633, 0.01172, 0.00345,
0.00406, 0.00528, 0.00772, 0.00616, 0.00677, 0.00355, 0.01104,
0.00521, 0.0147, 0.01105, 0.01209, 0.00504, 0.00948, 0.03351, 0.01053,
0.00731, 0.00853, 0.01941, 0.0047, 0.01097, 0.0148, 0.00514, 0.00314,
0.0088, 0.00558, 0.00619, 0.0048, 0.01246, 0.00663, 0.00602, 0.01046,
0.01168, 0.00402, 0.01873, 0.01351, 0.00507, 0.00568, 0.01517,
0.01778, 0.00873, 0.00612, 0.00229, 0.01056, 0.03476, 0.01283,
0.01161, 0.00456, 0.007, 0.00883, 0.00439, 0.01388, 0.00805, 0.00422,
0.03225, 0.05401, 0.00283, 0.00605, 0.00466, 0.00327, 0.0071, 0.0051,
0.00632, 0.0152, 0.0112, 0.00476, 0.01242, 0.00276, 0.00659, 0.01164,
0.00459, 0.00964, 0.00642, 0.00381, 0.00442, 0.00364, 0.00808,
0.00225, 0.00408, 0.01235, 0.01601, 0.01218, 0.00896, 0.00574,
0.00696, 0.0114, 0.02228, 0.00496, 0.00862, 0.02716, 0.01506, 0.0054,
0.05563, 0.00723, 0.00462, 0.01289, 0.01211, 0.01394, 0.00506,
0.00628, 0.0075, 0.02987, 0.00994, 0.01116, 0.0035, 0.00411, 0.00672,
0.00472, 0.01865, 0.00499, 0.01109, 0.00526, 0.01092, 0.00587,
0.00204, 0.00831, 0.02546, 0.00431, 0.04705, 0.00658, 0.0078, 0.01285,
0.00841, 0.01407, 0.00641, 0.00763, 0.0038, 0.02756, 0.00519, 0.02478,
0.00885, 0.00546, 0.01112, 0.00346, 0.0079, 0.014, 0.00373, 0.01966,
0.00495, 0.01505, 0.00278, 0.00661, 0.00339, 0.00844, 0.00461,
0.00905}

********************************************** MDVP:Shimmer
***********************************************

{0.06511, 0.09419, 0.0605, 0.04701, 0.01681, 0.02047, 0.02752,
0.01098, 0.02308, 0.05233, 0.01725, 0.04128, 0.0203, 0.01064, 0.02857,
0.01752, 0.01613, 0.05643, 0.01169, 0.0103, 0.02362, 0.01152, 0.01718,
0.01457, 0.03111, 0.03999, 0.02223, 0.09178, 0.01657, 0.02145,
0.01884, 0.01745, 0.04087, 0.01484, 0.04192, 0.02033, 0.03121,
0.06734, 0.0145, 0.04009, 0.02199, 0.03026, 0.01494, 0.06134, 0.03209,
0.03087, 0.01033, 0.01843, 0.0166, 0.05517, 0.01643, 0.06727, 0.01043,
0.04351, 0.04978, 0.03202, 0.04795, 0.02297, 0.03995, 0.01131,
0.01897, 0.0168, 0.01419, 0.01358, 0.01663, 0.03273, 0.02751, 0.00958,
0.01463, 0.02551, 0.01263, 0.06425, 0.02534, 0.01185, 0.02378,
0.03327, 0.04137, 0.01412, 0.03527, 0.01795, 0.01761, 0.013, 0.02293,
0.03198, 0.03381, 0.03886, 0.01022, 0.05279, 0.02215, 0.02093,
0.04374, 0.03225, 0.05384, 0.03852, 0.04313, 0.04879, 0.08143,
0.02442, 0.02503, 0.01659, 0.01015, 0.00954, 0.05428, 0.02286,
0.04479, 0.03235, 0.01259, 0.01642, 0.02791, 0.02852, 0.01503,
0.04689, 0.02008, 0.0717, 0.01242, 0.01564, 0.02574, 0.01608, 0.01791,
0.01469, 0.03767, 0.0617, 0.02296, 0.02662, 0.02018, 0.01574, 0.05492,
0.02184, 0.02645, 0.03716, 0.03272, 0.01279, 0.01906, 0.01706,
0.01201, 0.02838, 0.01828, 0.01567, 0.11908, 0.02682, 0.02177,
0.01472, 0.01194, 0.02343, 0.01516, 0.02143, 0.02448, 0.0176, 0.01299,
0.02126, 0.04024, 0.07959, 0.04912, 0.03658, 0.02814, 0.01909,
0.02719, 0.03485, 0.0419, 0.02536, 0.01831, 0.01919, 0.01997, 0.01458,
0.03156, 0.01851, 0.05925, 0.01346, 0.04932, 0.01024, 0.06725,
0.07118, 0.08684, 0.03044, 0.0381, 0.0583, 0.01756, 0.01495, 0.03715,
```

```
0.01861, 0.02427, 0.03759, 0.0664, 0.02105, 0.01966, 0.01644, 0.01444,
0.02498}

********************************************** MDVP:Shimmer(dB)
***********************************************

{0.482, 0.542, 0.135, 0.478, 0.129, 0.377, 0.361, 0.383, 0.381, 0.131,
0.125, 0.369, 0.111, 0.107, 0.103, 0.117, 0.283, 1.302, 0.267, 0.136,
0.431, 0.181, 0.21, 0.226, 0.097, 0.089, 0.228, 0.206, 0.216, 0.093,
0.224, 0.099, 0.476, 0.175, 0.659, 0.784, 0.212, 0.249, 0.124, 0.085,
0.618, 0.772, 0.833, 0.272, 0.442, 0.237, 0.426, 0.456, 0.186, 0.202,
0.327, 0.331, 0.313, 0.198, 0.307, 0.192, 0.891, 0.223, 0.571, 0.821,
0.18, 0.217, 0.379, 0.154, 0.158, 0.164, 0.168, 0.17, 0.152, 0.148,
0.142, 0.441, 0.236, 0.265, 0.339, 0.634, 0.171, 0.626, 0.584, 0.517,
0.14, 0.134, 0.255, 0.497, 0.263, 0.132, 0.35, 0.276, 0.93, 0.241,
0.112, 0.106, 0.116, 0.19, 0.235, 0.221, 0.094, 0.207, 0.209, 0.231,
0.233, 0.098, 0.09, 0.225, 0.348, 0.133, 0.422, 0.297, 0.281, 0.406,
0.246, 0.65, 0.275, 0.365, 0.58, 0.191, 0.308, 0.334, 0.328, 0.296,
0.197, 0.185, 0.435, 0.189, 0.342, 0.138, 0.257, 0.722, 0.126, 0.405,
0.325, 0.37, 0.143, 0.161, 0.151, 0.141, 0.145, 0.165, 0.155, 0.149,
0.163, 0.364, 0.438, 1.018, 0.483, 0.266, 0.637, 0.137, 0.256}

********************************************** Shimmer:APQ3
***********************************************

{0.02542, 0.0081, 0.01454, 0.00793, 0.03152, 0.03474, 0.01176,
0.01803, 0.01864, 0.02413, 0.04284, 0.00942, 0.00881, 0.0082, 0.01064,
0.0183, 0.00742, 0.01186, 0.00864, 0.0365, 0.01813, 0.02135, 0.03223,
0.00847, 0.00664, 0.00586, 0.00725, 0.00969, 0.00769, 0.01396,
0.01013, 0.00952, 0.04016, 0.01579, 0.00874, 0.02328, 0.01379,
0.00796, 0.01806, 0.00779, 0.01484, 0.01667, 0.01789, 0.0064, 0.01284,
0.01084, 0.01006, 0.05358, 0.01189, 0.00867, 0.01372, 0.00606,
0.00667, 0.00728, 0.02182, 0.05551, 0.01155, 0.00772, 0.01721,
0.01277, 0.00633, 0.0307, 0.00938, 0.01321, 0.00738, 0.00538, 0.0066,
0.00721, 0.01792, 0.03341, 0.01026, 0.02924, 0.02297, 0.00504,
0.00748, 0.0349, 0.01192, 0.01514, 0.02385, 0.00975, 0.03134, 0.02107,
0.02229, 0.02073, 0.03788, 0.01107, 0.02683, 0.00829, 0.01073, 0.0049,
0.00812, 0.02266, 0.01117, 0.015, 0.00534, 0.00656, 0.01483, 0.01805,
0.00839, 0.00883, 0.01205, 0.02032, 0.01771, 0.01432, 0.01371,
0.00849, 0.01154, 0.01659, 0.00754, 0.03357, 0.00476, 0.02896,
0.01547, 0.02757, 0.00703, 0.02679, 0.01713, 0.0093, 0.00469, 0.04421,
0.02062, 0.03611, 0.01035, 0.00974, 0.01235, 0.00774, 0.01279,
0.01079, 0.0134, 0.00696, 0.02228, 0.00757, 0.03804, 0.01323, 0.00557,
0.00679, 0.02055, 0.02699, 0.00906, 0.01289, 0.00967, 0.02865,
0.00889, 0.01394, 0.0095, 0.02021, 0.01638, 0.02587, 0.00855, 0.0136,
0.02187, 0.01604, 0.00777, 0.00455, 0.01143, 0.00882, 0.01265,
0.02336, 0.00726, 0.01475, 0.00631, 0.01441, 0.00614, 0.01241,
0.01424, 0.01868, 0.00563, 0.01773, 0.01268, 0.01373, 0.03671,
0.05647, 0.00468, 0.02383, 0.02749, 0.03515, 0.02471, 0.0261, 0.01644,
0.00617, 0.00861, 0.01732, 0.00522, 0.0141}
```

```
********************************************* Shimmer:APQ5
*********************************************

{0.04518, 0.04962, 0.00888, 0.02498, 0.00932, 0.0061, 0.01759,
0.04101, 0.0263, 0.0142, 0.00776, 0.01342, 0.00898, 0.00959, 0.00576,
0.01725, 0.01325, 0.0082, 0.01003, 0.00925, 0.01108, 0.01169, 0.0123,
0.00786, 0.03572, 0.0063, 0.05426, 0.01657, 0.01057, 0.01223, 0.00901,
0.02494, 0.01284, 0.00701, 0.0165, 0.02521, 0.00606, 0.00789, 0.00972,
0.02321, 0.03714, 0.01033, 0.01277, 0.01399, 0.0247, 0.01199, 0.02592,
0.01582, 0.01121, 0.01321, 0.01426, 0.01365, 0.00721, 0.01609,
0.01992, 0.00582, 0.00948, 0.05005, 0.01375, 0.02768, 0.01558,
0.01219, 0.01341, 0.00941, 0.0068, 0.04005, 0.00802, 0.0794, 0.03022,
0.01812, 0.02422, 0.01012, 0.02161, 0.04825, 0.02466, 0.01439,
0.01117, 0.01483, 0.01805, 0.00717, 0.01161, 0.02493, 0.04791,
0.01405, 0.01144, 0.01893, 0.02415, 0.00744, 0.01815, 0.03347, 0.0353,
0.02076, 0.00788, 0.00971, 0.01859, 0.00588, 0.00832, 0.0313, 0.00632,
0.01459, 0.00815, 0.00937, 0.02591, 0.01886, 0.01964, 0.01581,
0.00876, 0.01625, 0.00659, 0.01103, 0.02374, 0.01347, 0.02174, 0.0254,
0.00825, 0.01974, 0.00886, 0.00747, 0.01191, 0.01574, 0.0073, 0.01296,
0.04282, 0.03794, 0.03672, 0.02567, 0.0194, 0.02245, 0.00957, 0.00879,
0.01906, 0.00818, 0.04265, 0.01062, 0.03526, 0.01994, 0.01289,
0.01272, 0.0095, 0.01072, 0.0458, 0.00933, 0.01177, 0.00977, 0.03858,
0.0116, 0.01038, 0.01421, 0.01343, 0.03963, 0.05556, 0.01804, 0.02231,
0.01909, 0.0076, 0.01021, 0.00621, 0.01553, 0.02302, 0.01075, 0.01841,
0.00631, 0.0057, 0.01258, 0.03112, 0.01641, 0.02024, 0.0158, 0.02973,
0.01058, 0.02451, 0.00641, 0.01024, 0.00946, 0.00885, 0.04254,
0.01495, 0.0099, 0.02383, 0.01478, 0.01783, 0.04159, 0.01722, 0.01017,
0.00956, 0.012, 0.04003, 0.0181, 0.00905}

********************************************* MDVP:APQ
*********************************************

{0.0343, 0.01149, 0.01715, 0.13778, 0.01271, 0.02603, 0.00871,
0.00993, 0.03091, 0.01559, 0.01359, 0.00915, 0.0646, 0.03772, 0.02074,
0.01491, 0.01691, 0.01552, 0.0123, 0.02301, 0.02745, 0.02084, 0.01318,
0.01179, 0.02067, 0.05114, 0.01301, 0.02006, 0.01345, 0.02877,
0.01667, 0.00762, 0.03243, 0.01267, 0.00928, 0.01433, 0.01799,
0.01233, 0.01677, 0.03392, 0.01033, 0.02704, 0.04802, 0.0569, 0.02809,
0.04114, 0.02931, 0.01016, 0.04368, 0.04246, 0.01826, 0.02148,
0.01382, 0.0086, 0.01948, 0.01104, 0.01931, 0.02802, 0.01148, 0.01009,
0.01331, 0.01636, 0.04683, 0.04134, 0.08808, 0.02402, 0.01758,
0.01497, 0.01263, 0.0431, 0.00802, 0.06259, 0.02073, 0.01246, 0.01307,
0.01612, 0.01751, 0.02056, 0.01351, 0.0351, 0.04398, 0.01151, 0.01734,
0.00951, 0.01717, 0.01256, 0.02971, 0.01944, 0.01439, 0.02571,
0.02876, 0.05174, 0.01344, 0.01666, 0.01144, 0.01588, 0.01771,
0.05767, 0.02137, 0.02259, 0.0131, 0.02764, 0.03635, 0.0172, 0.0253,
0.01059, 0.01947, 0.01852, 0.00903, 0.01652, 0.02157, 0.02784,
0.02018, 0.04055, 0.03455, 0.06824, 0.01879, 0.02428, 0.00957, 0.0134,
0.04465, 0.0114, 0.03316, 0.02916, 0.02455, 0.01767, 0.01506, 0.01367,
```

```
0.01872, 0.0437, 0.01133, 0.03736, 0.00811, 0.01194, 0.01255, 0.0277,
0.01621, 0.0378, 0.01604, 0.01909, 0.00882, 0.01892, 0.02214, 0.01831,
0.05783, 0.04451, 0.01309, 0.02519, 0.00726, 0.0359, 0.01797, 0.01614,
0.03651, 0.02824, 0.06359, 0.01014, 0.03051, 0.01075, 0.01685, 0.0219,
0.01363, 0.00719, 0.08318, 0.01285, 0.0119, 0.01251, 0.01956, 0.02139,
0.01312, 0.03105, 0.01756, 0.01373, 0.02444, 0.01234, 0.02949,
0.03088, 0.01095, 0.04525, 0.01661, 0.06196, 0.04464, 0.014, 0.012,
0.02454, 0.03908, 0.06023, 0.0322, 0.01366, 0.01949}

********************************************* Shimmer:DDA
**********************************************

{0.0827, 0.03969, 0.06799, 0.11012, 0.06406, 0.01471, 0.02542,
0.02925, 0.01898, 0.02908, 0.02647, 0.02308, 0.01603, 0.01403, 0.0445,
0.06321, 0.10422, 0.03867, 0.06165, 0.02518, 0.01979, 0.02623,
0.06985, 0.13262, 0.08595, 0.05592, 0.0265, 0.02789, 0.06097, 0.06219,
0.02389, 0.04426, 0.01406, 0.04914, 0.02921, 0.08771, 0.06185,
0.07761, 0.03104, 0.02643, 0.0246, 0.02321, 0.05368, 0.0549, 0.04114,
0.04019, 0.03836, 0.02226, 0.02487, 0.02548, 0.04324, 0.09669,
0.03253, 0.02592, 0.0227, 0.03463, 0.03724, 0.05439, 0.01992, 0.02436,
0.02175, 0.04812, 0.01758, 0.03429, 0.03568, 0.02602, 0.05605,
0.07625, 0.07154, 0.11411, 0.02307, 0.02429, 0.01968, 0.04188,
0.03039, 0.09455, 0.0532, 0.16654, 0.08689, 0.09211, 0.08096, 0.16942,
0.07008, 0.05164, 0.04137, 0.02666, 0.0412, 0.05408, 0.11363, 0.0715,
0.03615, 0.02849, 0.06062, 0.0332, 0.0211, 0.03964, 0.02337, 0.07238,
0.03852, 0.10024, 0.16074, 0.10546, 0.12851, 0.03191, 0.02164,
0.04079, 0.03557, 0.03218, 0.0233, 0.05377, 0.06892, 0.01364, 0.04933,
0.03529, 0.04272, 0.03706, 0.04977, 0.01513, 0.02184, 0.10949,
0.01818, 0.04543, 0.12047, 0.0335, 0.04499, 0.02001, 0.02245, 0.03794,
0.05414, 0.02855, 0.02089, 0.03804, 0.01689, 0.01567, 0.01428,
0.02316, 0.06685, 0.05197, 0.03831, 0.04736, 0.04231, 0.01672,
0.02177, 0.02038, 0.04641, 0.09403, 0.02814, 0.08037, 0.05312,
0.06688, 0.04363, 0.02214, 0.02719, 0.05417, 0.04451, 0.05139,
0.01892, 0.02902, 0.05, 0.0238, 0.01919, 0.04295, 0.08247, 0.03851,
0.10833, 0.01614, 0.01841, 0.0783, 0.04322, 0.04183, 0.03017, 0.03078,
0.07413, 0.01407, 0.01851, 0.02583, 0.04115, 0.02566, 0.02827,
0.02261, 0.06545, 0.1047, 0.06562, 0.02488, 0.1007, 0.03576, 0.0805,
0.03237, 0.03464}

********************************************* NHR
**********************************************

{0.0222, 0.0202, 0.00488, 0.00871, 0.04179, 0.01237, 0.00454, 0.0062,
0.00681, 0.0042, 0.00481, 0.01813, 0.01874, 0.00786, 0.00586, 0.00969,
0.01796, 0.0083, 0.0184, 0.0547, 0.08151, 0.16265, 0.0043, 0.01823,
0.01179, 0.00474, 0.00135, 0.01728, 0.00257, 0.03365, 0.01328,
0.02599, 0.00301, 0.00623, 0.00484, 0.02782, 0.02887, 0.0085, 0.00955,
0.00233, 0.00677, 0.00965, 0.00704, 0.01914, 0.00504, 0.0107, 0.00243,
0.10323, 0.0369, 0.00487, 0.08069, 0.00609, 0.02663, 0.02707, 0.01036,
0.03151, 0.01802, 0.01724, 0.02629, 0.0128, 0.01141, 0.01968, 0.0181,
```

```
0.0068, 0.03361, 0.0753, 0.02073, 0.04398, 0.01778, 0.07889, 0.00351,
0.00856, 0.00595, 0.01222, 0.00839, 0.01161, 0.01022, 0.01849,
0.00578, 0.00639, 0.02659, 0.003, 0.31482, 0.2593, 0.01049, 0.00344,
0.01554, 0.01493, 0.01859, 0.0091, 0.02764, 0.01015, 0.03191, 0.01337,
0.00432, 0.01825, 0.00737, 0.00415, 0.00476, 0.04611, 0.00659,
0.00903, 0.00581, 0.00703, 0.00442, 0.00947, 0.03828, 0.01435,
0.04882, 0.00852, 0.04238, 0.10715, 0.01018, 0.0235, 0.03882, 0.00435,
0.01062, 0.01767, 0.02211, 0.0074, 0.00479, 0.0054, 0.0034, 0.0316,
0.04214, 0.00462, 0.00401, 0.01211, 0.01794, 0.0161, 0.04824, 0.06051,
0.04441, 0.01316, 0.00167, 0.00611, 0.03031, 0.00472, 0.00533,
0.02431, 0.08725, 0.10952, 0.00072, 0.21713, 0.02631, 0.01143,
0.00882, 0.02919, 0.00238, 0.01309, 0.01048, 0.0117, 0.01353, 0.00265,
0.11843, 0.00065, 0.01658, 0.02485, 0.0057, 0.01397, 0.02529, 0.00231,
0.00675, 0.0118, 0.01041, 0.01929, 0.16744, 0.00119, 0.03871, 0.02278,
0.00607, 0.02183, 0.0059, 0.01095, 0.07223, 0.06057, 0.10748, 0.0281,
0.00495, 0.01827, 0.0201, 0.00478, 0.00339, 0.01166, 0.00905}

********************************************** HNR
***********************************************

{20.366, 8.867, 8.441, 10.489, 11.866, 11.744, 13.893, 13.922, 15.924,
16.176, 17.28, 17.153, 19.649, 20.376, 20.644, 22.333, 19.085, 24.889,
25.856, 26.738, 26.31, 26.822, 26.453, 26.805, 26.138, 26.415, 25.135,
25.438, 26.892, 30.94, 30.775, 32.684, 33.047, 31.732, 23.389, 24.581,
18.954, 18.54, 19.368, 25.023, 19.651, 15.433, 16.747, 17.536, 17.707,
17.366, 18.57, 18.78, 18.305, 18.784, 18.857, 19.979, 19.055, 19.659,
19.664, 18.33, 20.338, 24.178, 20.536, 20.969, 20.68, 21.033, 21.378,
21.414, 21.028, 21.875, 22.817, 22.603, 22.244, 22.219, 21.305,
23.216, 23.162, 23.671, 23.683, 22.866, 24.692, 24.951, 21.824,
19.075, 24.412, 25.908, 25.119, 25.03, 25.964, 25.97, 26.775, 25.82,
26.759, 26.833, 26.356, 27.421, 26.842, 26.369, 26.436, 26.143,
28.409, 21.209, 19.517, 24.679, 23.949, 9.449, 23.079, 24.151, 24.199,
21.864, 22.431, 23.133, 23.008, 19.147, 25.445, 19.02, 21.534, 25.619,
21.52, 21.02, 12.529, 14.739, 14.367, 14.989, 15.648, 23.693, 21.083,
17.06, 17.883, 21.934, 18.178, 18.801, 18.702, 18.687, 19.493, 19.56,
19.2, 19.196, 19.013, 20.651, 20.422, 20.767, 20.264, 22.953, 21.66,
21.718, 21.812, 21.862, 21.118, 22.468, 22.066, 22.736, 21.422,
21.693, 23.831, 23.145, 23.37, 22.663, 23.239, 24.922, 24.886, 24.547,
24.602, 24.971, 25.703, 25.175, 25.197, 25.429, 25.554, 25.742,
25.032, 25.368, 25.678, 25.69, 27.166, 26.017, 26.55, 26.547, 26.005,
23.958, 24.775, 29.746, 29.928, 24.078, 19.388, 25.02, 21.219, 12.435,
12.359, 12.298, 18.447, 15.338, 15.06, 19.269, 22.407, 22.085, 20.437,
21.528, 24.743}

********************************************** status
***********************************************

{0, 1}

********************************************** RPDE
```

```
*********************************************

{0.615551, 0.547037, 0.56738, 0.553134, 0.543299, 0.306443, 0.540049,
0.558586, 0.495954, 0.530529, 0.536009, 0.637518, 0.528485, 0.482296,
0.46716, 0.522812, 0.407701, 0.6283, 0.653139, 0.469928, 0.611137,
0.358773, 0.371837, 0.402591, 0.630547, 0.398499, 0.623209, 0.677131,
0.635015, 0.487756, 0.311369, 0.457541, 0.447285, 0.462516, 0.565924,
0.486738, 0.39661, 0.397937, 0.538016, 0.65168, 0.32648, 0.360148,
0.611679, 0.583574, 0.55287, 0.639808, 0.640945, 0.555303, 0.364867,
0.644954, 0.489345, 0.459766, 0.434969, 0.458359, 0.470478, 0.25657,
0.474791, 0.422229, 0.305062, 0.420383, 0.498133, 0.487407, 0.576644,
0.606273, 0.56161, 0.36909, 0.618663, 0.507826, 0.334171, 0.478024,
0.408598, 0.566424, 0.584164, 0.508479, 0.594387, 0.547975, 0.436084,
0.431674, 0.296888, 0.344252, 0.507504, 0.384868, 0.660125, 0.589956,
0.447456, 0.263654, 0.671378, 0.468621, 0.352396, 0.538688, 0.638545,
0.454444, 0.590951, 0.541781, 0.366329, 0.393563, 0.544805, 0.55461,
0.501037, 0.400088, 0.414783, 0.429895, 0.59604, 0.63742, 0.447979,
0.58339, 0.522746, 0.403884, 0.579597, 0.4606, 0.463514, 0.602874,
0.6479, 0.413295, 0.536102, 0.509127, 0.331508, 0.437031, 0.665318,
0.596362, 0.470422, 0.497554, 0.462803, 0.625362, 0.440988, 0.631099,
0.598515, 0.372222, 0.663842, 0.341435, 0.637814, 0.467489, 0.448439,
0.432439, 0.429484, 0.50238, 0.557045, 0.387482, 0.649554, 0.625866,
0.566867, 0.305429, 0.577802, 0.427785, 0.417356, 0.415564, 0.430166,
0.460139, 0.368535, 0.27685, 0.489538, 0.457702, 0.418622, 0.340068,
0.629574, 0.576084, 0.603515, 0.470972, 0.380253, 0.599371, 0.654945,
0.49748, 0.671299, 0.338097, 0.566849, 0.685151, 0.438296, 0.341788,
0.598714, 0.57101, 0.624811, 0.606344, 0.427627, 0.361232, 0.629017,
0.61906, 0.365488, 0.441097, 0.65341, 0.431285, 0.491345, 0.450798,
0.556494, 0.653427, 0.465946, 0.396793, 0.537264, 0.585169, 0.329577,
0.610367, 0.451221, 0.498877, 0.405991, 0.407567, 0.513237}

********************************************* DFA
*********************************************

{0.815285, 0.825288, 0.825069, 0.773587, 0.778834, 0.681811, 0.731444,
0.819032, 0.679834, 0.686264, 0.631653, 0.787896, 0.67793, 0.715121,
0.632631, 0.656516, 0.772416, 0.79252, 0.635285, 0.760361, 0.811843,
0.68413, 0.670475, 0.725216, 0.646818, 0.722254, 0.659132, 0.733659,
0.663884, 0.643327, 0.735546, 0.623731, 0.605417, 0.817756, 0.778747,
0.74148, 0.76832, 0.62671, 0.644692, 0.740837, 0.712466, 0.738245,
0.654331, 0.720916, 0.742133, 0.743937, 0.664357, 0.768845, 0.667654,
0.686894, 0.727313, 0.706687, 0.776158, 0.785714, 0.778349, 0.669565,
0.762959, 0.674953, 0.696049, 0.7567, 0.761255, 0.766204, 0.673636,
0.693429, 0.652025, 0.742737, 0.754073, 0.712199, 0.723096, 0.763242,
0.75718, 0.635204, 0.655239, 0.744064, 0.775708, 0.675865, 0.674562,
0.726652, 0.720908, 0.807217, 0.722356, 0.701404, 0.704087, 0.634267,
0.789532, 0.723797, 0.698951, 0.738703, 0.733232, 0.754449, 0.728421,
0.727747, 0.684373, 0.676258, 0.694399, 0.735136, 0.627337, 0.745957,
0.742055, 0.729067, 0.719974, 0.699787, 0.721308, 0.68608, 0.762508,
0.776156, 0.730387, 0.815908, 0.789977, 0.732903, 0.724045, 0.708617,
```

```
0.574282, 0.659333, 0.7667, 0.819521, 0.823484, 0.819235, 0.764112,
0.763262, 0.798463, 0.719467, 0.783626, 0.766209, 0.638928, 0.646846,
0.700246, 0.722085, 0.714485, 0.694571, 0.655683, 0.733165, 0.705658,
0.759203, 0.737948, 0.628058, 0.643956, 0.665833, 0.654027, 0.71436,
0.727863, 0.654172, 0.821364, 0.653823, 0.718839, 0.770466, 0.75932,
0.813432, 0.676066, 0.58271, 0.646786, 0.719576, 0.817396, 0.790117,
0.768974, 0.683296, 0.775933, 0.683244, 0.740539, 0.646167, 0.756482,
0.758324, 0.745526, 0.690892, 0.691483, 0.662668, 0.656846, 0.641418,
0.732479, 0.765623, 0.673086, 0.729586, 0.69779, 0.734504, 0.656182,
0.658245, 0.708144, 0.764036, 0.676023, 0.665945, 0.762726, 0.628232,
0.683761, 0.678874, 0.71217, 0.736964, 0.685057, 0.657899, 0.741367,
0.81634, 0.779612, 0.661735, 0.630409, 0.793509, 0.741899}

********************************************** spread1
***********************************************

{-5.410336, -5.966779, -5.498456, -3.760348, -7.964984, -7.31951, -
6.006647, -6.966321, -6.149653, -6.647379, -6.452058, -5.571843, -
5.207985, -5.657899, -5.952058, -5.540351, -4.075192, -4.443179, -
4.330956, -4.913885, -5.825257, -3.700544, -6.538586, -6.448134, -
6.311987, -6.195325, -5.585259, -2.839756, -6.710219, -7.293801, -
7.3483, -7.067931, -7.695734, -7.044105, -6.261141, -6.99582, -
6.112667, -5.185987, -6.085567, -5.477592, -5.418787, -5.44514, -
5.301321, -5.456811, -4.60826, -4.876336, -4.711007, -5.704053, -
5.592584, -3.297668, -3.583722, -5.61907, -6.016891, -7.040508, -
7.24562, -6.18359, -6.025367, -6.152551, -6.878393, -6.323531, -
5.660217, -5.720868, -5.31336, -5.132032, -5.634576, -4.438453, -
4.20273, -5.898673, -4.865194, -5.617124, -6.186128, -6.744577, -
7.496264, -7.057869, -7.517934, -7.169701, -6.650471, -6.793547, -
6.366916, -6.729713, -6.051233, -5.409423, -5.86975, -5.775966, -
5.557447, -5.44004, -4.242867, -4.649573, -5.391029, -5.022288, -
5.288912, -4.020042, -5.944191, -4.378916, -4.276605, -4.892495, -
3.444478, -5.529833, -6.486822, -6.890021, -6.787197, -7.3045, -
6.934474, -6.439398, -6.635729, -6.836811, -6.411497, -6.251425, -
5.340115, -5.634322, -5.389129, -5.845099, -5.25832, -4.476755, -
4.117501, -4.379411, -4.673241, -4.796845, -4.484303, -4.239028, -
2.434031, -2.929379, -6.012559, -6.816086, -3.949079, -5.736781, -
7.314237, -7.682587, -7.777685, -6.482096, -6.105098, -6.997403, -
6.981201, -6.471427, -5.498678, -5.24977, -5.324574, -5.159169, -
5.892061, -4.554466, -4.095442, -4.508984, -5.4351, -3.269487, -
6.132663, -5.943501, -6.27717, -5.070096, -5.783272, -7.31055, -
7.156076, -7.111576, -7.018057, -6.27169, -6.006414, -6.41744, -
6.261446, -6.359018, -5.18696, -5.248776, -5.515071, -5.96304, -
5.333619, -4.333543, -4.441519, -4.597834, -4.913137, -4.654894, -
3.377325, -5.866357, -6.135296, -5.594275, -7.120925, -6.431119, -
6.167603, -6.547148, -6.247076, -6.600023, -6.739151, -5.011879, -
5.115212, -5.517173, -5.436135, -5.283009, -4.960234, -4.609161, -
5.711205, -5.198864, -5.724056, -7.072419, -6.689151, -5.79182, -
4.813031, -3.747787, -2.93107}
```

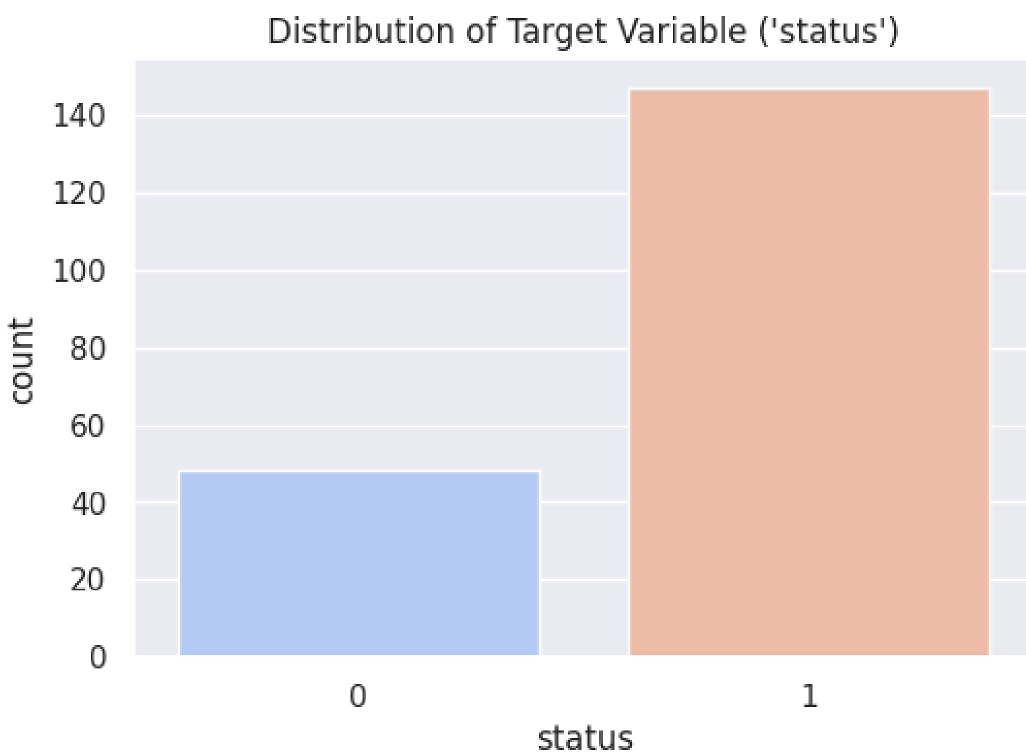
```
********************************************** spread2
**********************************************

{0.327769, 0.434326, 0.237622, 0.221711, 0.196102, 0.221727, 0.262564,
0.173319, 0.066994, 0.22685, 0.17352, 0.281618, 0.320385, 0.310746,
0.19215, 0.180528, 0.389232, 0.279933, 0.249172, 0.158453, 0.254909,
0.089267, 0.216204, 0.15331, 0.265315, 0.234513, 0.236853, 0.226278,
0.212294, 0.170183, 0.163519, 0.212054, 0.142466, 0.20363, 0.230532,
0.229298, 0.220617, 0.197938, 0.291954, 0.116636, 0.190667, 0.018689,
0.206768, 0.133917, 0.12795, 0.152941, 0.129303, 0.171088, 0.178713,
0.304107, 0.335357, 0.24275, 0.311173, 0.334147, 0.257682, 0.325996,
0.266392, 0.262384, 0.264967, 0.269866, 0.20566, 0.250572, 0.322044,
0.372114, 0.27328, 0.217013, 0.250283, 0.175691, 0.102083, 0.242861,
0.087165, 0.160414, 0.306014, 0.091608, 0.17854, 0.205558, 0.08784,
0.14478, 0.202146, 0.195976, 0.213353, 0.414758, 0.17227, 0.183218,
0.197919, 0.375531, 0.261549, 0.109397, 0.260481, 0.238298, 0.056844,
0.149694, 0.006274, 0.226528, 0.266482, 0.299111, 0.210279, 0.209866,
0.254498, 0.151814, 0.254989, 0.262633, 0.306636, 0.086372, 0.15989,
0.450493, 0.256454, 0.393056, 0.188056, 0.259229, 0.207454, 0.160267,
0.203653, 0.239764, 0.152813, 0.200873, 0.196371, 0.146948, 0.115697,
0.234196, 0.164529, 0.073298, 0.389295, 0.182459, 0.280091, 0.155097,
0.098648, 0.120956, 0.343073, 0.335753, 0.266892, 0.249494, 0.300067,
0.355736, 0.396746, 0.158266, 0.121952, 0.35787, 0.160686, 0.220434,
0.33559, 0.391002, 0.329066, 0.340176, 0.201861, 0.210185, 0.279789,
0.290024, 0.341169, 0.315074, 0.18455, 0.262281, 0.265699, 0.23307,
0.288917, 0.207156, 0.270641, 0.143359, 0.299793, 0.192375, 0.310163,
0.078202, 0.397749, 0.356224, 0.218037, 0.201095, 0.175181, 0.15883,
0.183721, 0.22089, 0.177551, 0.127642, 0.063412, 0.095882, 0.176316,
0.240875, 0.299549, 0.184378, 0.207914, 0.194627, 0.303214, 0.363566,
0.315903, 0.109256, 0.191576, 0.278679, 0.181701, 0.224852, 0.218885,
0.345238, 0.184896, 0.340256, 0.246404, 0.210458}

********************************************** D2
**********************************************

{1.854785, 2.301442, 2.486855, 2.405554, 2.33218, 2.342259, 2.18756,
2.064693, 2.322511, 2.432792, 2.407313, 2.856676, 2.3658, 1.777901,
2.034827, 2.576563, 2.964568, 2.209021, 2.784312, 2.679772, 1.852542,
2.536527, 2.473239, 2.502336, 2.539724, 2.375138, 3.13655, 2.672362,
2.840556, 3.413649, 2.235197, 1.831691, 2.557536, 2.017753, 2.428306,
2.344876, 2.441612, 3.098256, 2.690917, 2.090438, 2.465528, 2.45113,
2.151121, 2.560422, 1.872946, 1.871871, 2.519422, 2.416838, 2.938114,
2.816781, 2.655744, 3.671155, 3.317586, 2.910213, 2.8923, 2.88245,
1.91399, 2.205546, 2.314209, 2.32156, 2.007923, 2.277927, 2.344336,
2.080121, 2.671825, 2.499148, 2.489191, 3.274865, 1.852402, 1.889002,
2.256699, 2.278687, 2.645959, 2.143851, 2.296873, 3.10901, 2.68624,
2.445502, 2.419253, 2.138608, 1.972297, 1.974857, 1.957961, 2.447064,
2.225815, 2.003032, 2.638279, 2.702355, 2.665133, 2.095237, 2.449763,
2.161936, 2.152083, 1.92294, 1.862092, 2.103106, 2.361532, 2.991063,
2.376749, 2.699645, 3.099301, 3.007096, 3.079221, 2.679185, 1.881767,
```

```
2.439597, 2.692176, 2.079922, 2.118596, 2.359973, 2.398422, 2.344348,
2.640798, 2.527742, 2.445646, 2.470746, 1.827012, 1.994387, 1.996146,
2.125618, 2.916777, 2.477082, 2.137075, 2.608749, 2.01353, 2.316346,
2.555477, 2.642276, 2.054419, 2.264501, 2.846369, 2.589702, 2.269398,
2.382544, 2.374073, 2.330716, 2.223719, 2.108873, 1.512275, 2.642476,
2.73971, 2.241742, 2.975889, 2.925862, 2.193412, 1.423287, 2.291558,
2.205024, 2.232576, 2.631793, 2.963799, 2.103014, 2.266432, 2.120412,
1.743867, 1.929715, 2.408689, 1.544609, 1.821297, 2.027228, 2.431854,
2.654271, 2.065477, 2.174306, 2.958815, 3.007463, 2.004719, 2.251553,
2.058658, 2.657476, 1.986899, 1.840198, 2.041277, 2.021591, 2.634633,
2.004055, 2.550961, 2.328513, 2.442906, 2.129924, 2.845109, 2.264226,
2.028612, 2.498224, 2.118496, 2.460791, 2.547508, 2.392122, 2.014606,
2.053601, 2.51632, 3.142364, 3.184027, 1.928708, 1.765957}

********************************************** PPE
***********************************************

{0.377429, 0.316395, 0.3167, 0.259765, 0.095032, 0.105993, 0.108514,
0.085569, 0.093534, 0.365391, 0.301952, 0.277227, 0.214346, 0.164916,
0.183572, 0.056141, 0.270173, 0.220546, 0.216638, 0.170633, 0.103224,
0.113942, 0.209863, 0.112878, 0.115515, 0.378483, 0.220657, 0.19273,
0.086398, 0.119308, 0.231571, 0.138512, 0.199889, 0.141929, 0.242119,
0.160809, 0.229892, 0.24308, 0.244948, 0.169923, 0.215724, 0.121777,
0.231723, 0.09632, 0.102706, 0.11513, 0.13639, 0.277948, 0.251972,
0.093193, 0.160376, 0.174152, 0.214075, 0.165827, 0.274407, 0.284654,
0.368674, 0.357775, 0.275931, 0.228624, 0.234589, 0.430788, 0.09147,
0.098555, 0.147403, 0.180828, 0.327978, 0.160812, 0.05761, 0.09622,
0.185668, 0.209191, 0.247455, 0.212386, 0.268144, 0.370961, 0.454721,
0.1701, 0.068501, 0.160691, 0.225461, 0.186489, 0.215558, 0.222716,
0.196535, 0.228319, 0.340623, 0.179677, 0.28278, 0.418646, 0.131728,
0.367233, 0.144105, 0.073581, 0.091546, 0.105306, 0.141422, 0.132703,
0.219514, 0.249703, 0.261305, 0.189032, 0.168895, 0.332634, 0.410335,
0.271362, 0.253556, 0.128872, 0.162999, 0.11573, 0.344834, 0.242981,
0.260015, 0.112838, 0.163118, 0.301487, 0.044539, 0.260375, 0.206256,
0.238281, 0.457533, 0.170106, 0.177807, 0.119652, 0.184985, 0.194052,
0.23252, 0.226247, 0.173218, 0.097336, 0.163755, 0.24974, 0.215961,
0.135242, 0.18558, 0.177275, 0.151709, 0.274387, 0.19771, 0.444774,
0.232861, 0.336085, 0.168581, 0.22052, 0.156368, 0.106802, 0.112856,
0.075587, 0.120605, 0.138868, 0.335041, 0.260633, 0.259451, 0.101516,
0.368975, 0.211756, 0.322111, 0.314464, 0.285695, 0.326197, 0.244512,
0.130554, 0.232744, 0.264666, 0.527367, 0.174429, 0.184067, 0.181988,
0.356881, 0.100881, 0.232209, 0.220968, 0.117399, 0.141958, 0.147491,
0.193918, 0.218164, 0.226156, 0.18818, 0.200423, 0.144614, 0.13412,
0.13305, 0.103561, 0.152428, 0.091604, 0.133867, 0.123306, 0.332086,
0.234809, 0.160306, 0.202879, 0.252404, 0.148569, 0.159777}
```

According to above results there are no any unnecessary characters in the dataset that would effect the performance of the model

Checking how the dependent variable varies(Balanced/Imbalanced)

```
# Distribution of 'status'
print("\nValue Counts of 'status':")
print(df['status'].value_counts())
plt.figure(figsize=(6, 4))
sns.countplot(x='status', data=df, palette='coolwarm')
plt.title("Distribution of Target Variable ('status')")
plt.show()


Value Counts of 'status':
status
1    147
0     48
Name: count, dtype: int64
```



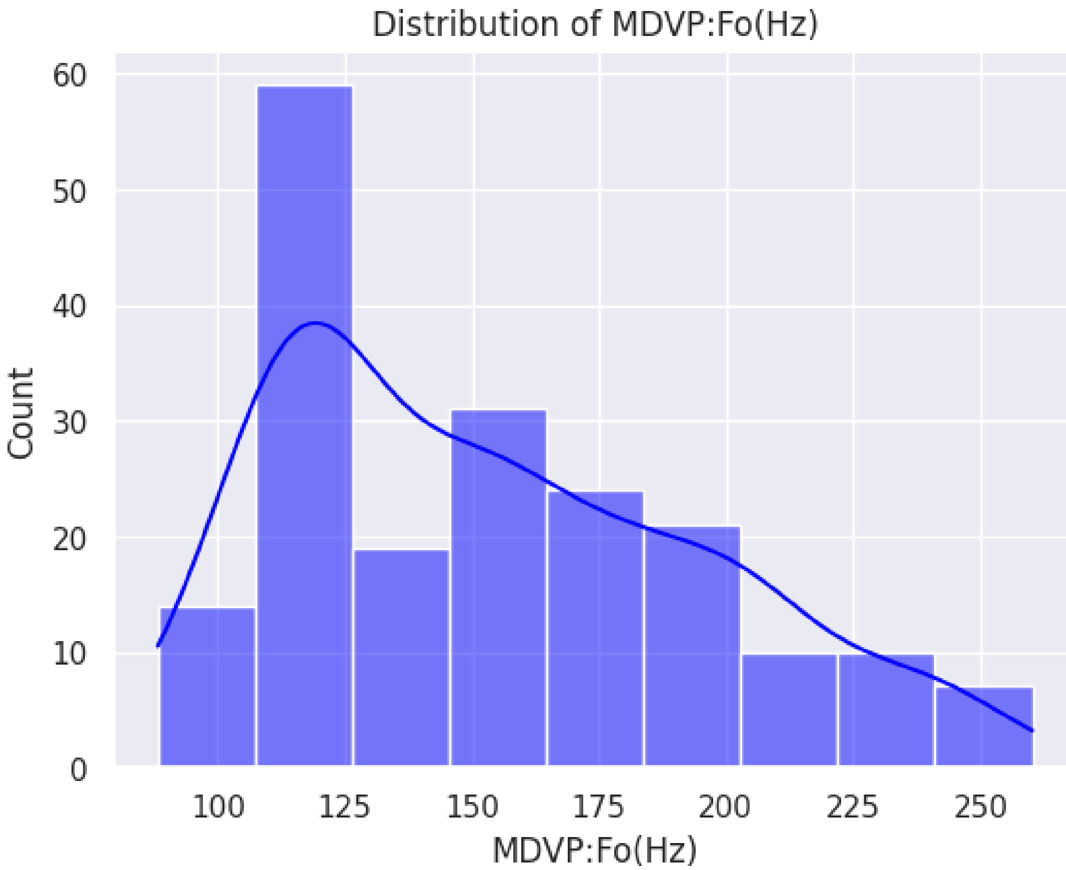
This results suggest that we need to scale the variables to avoid bias toward higher class

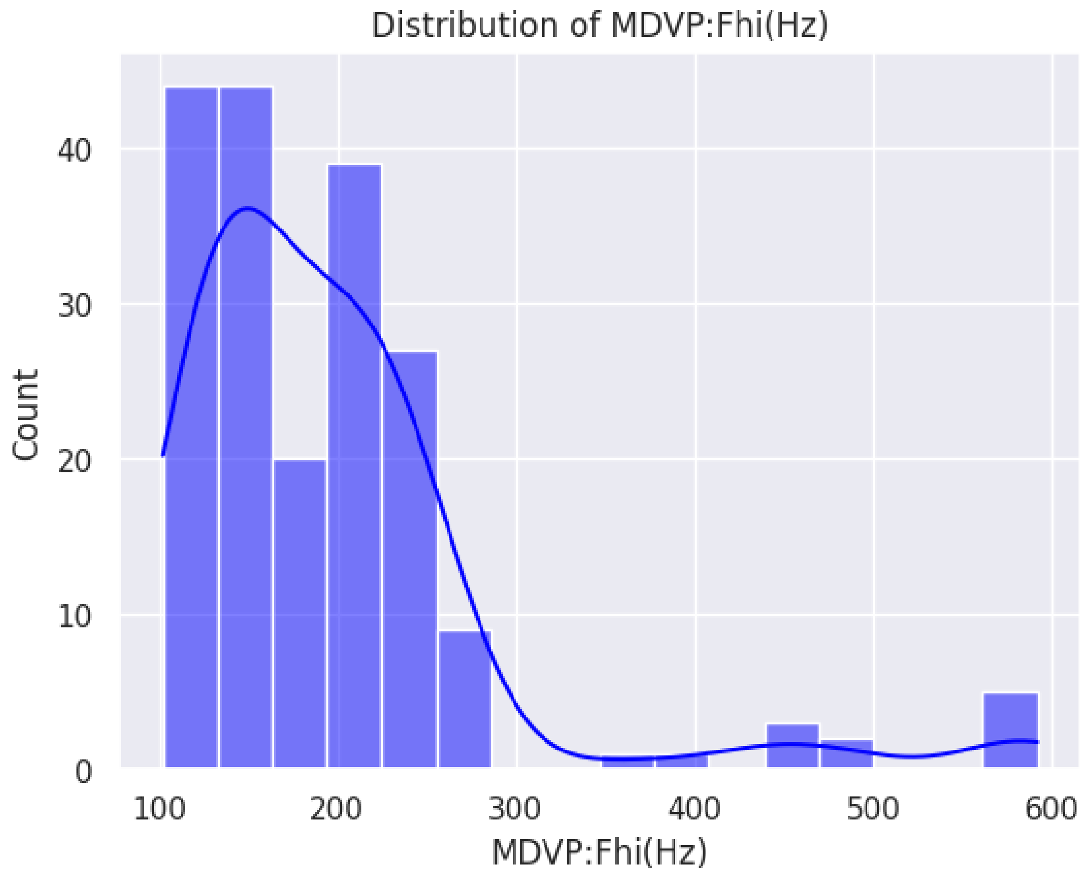
Finding the distribution of the dataset

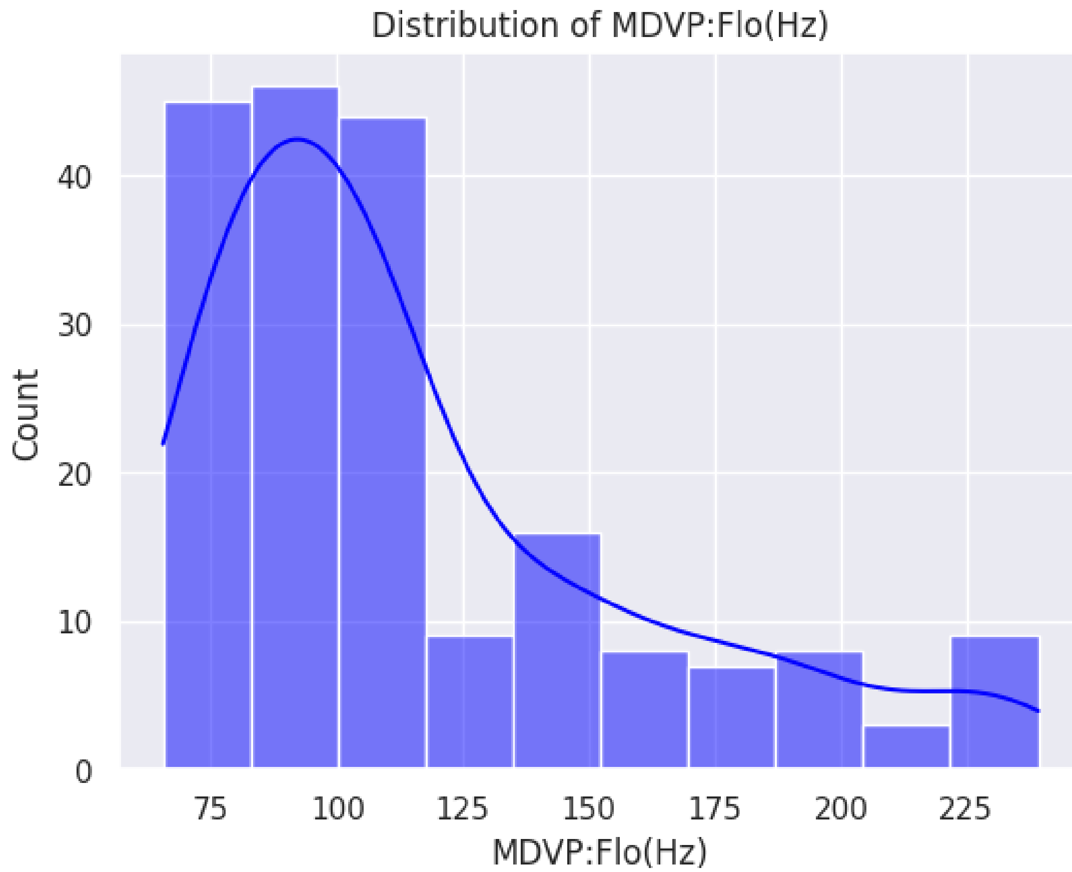
```
# Distribution plots for all numeric columns
print("\nDistribution plots for numeric features:")
for col in df.select_dtypes(include=['number']).columns:
    sns.histplot(df[col], kde=True, color='blue')
```

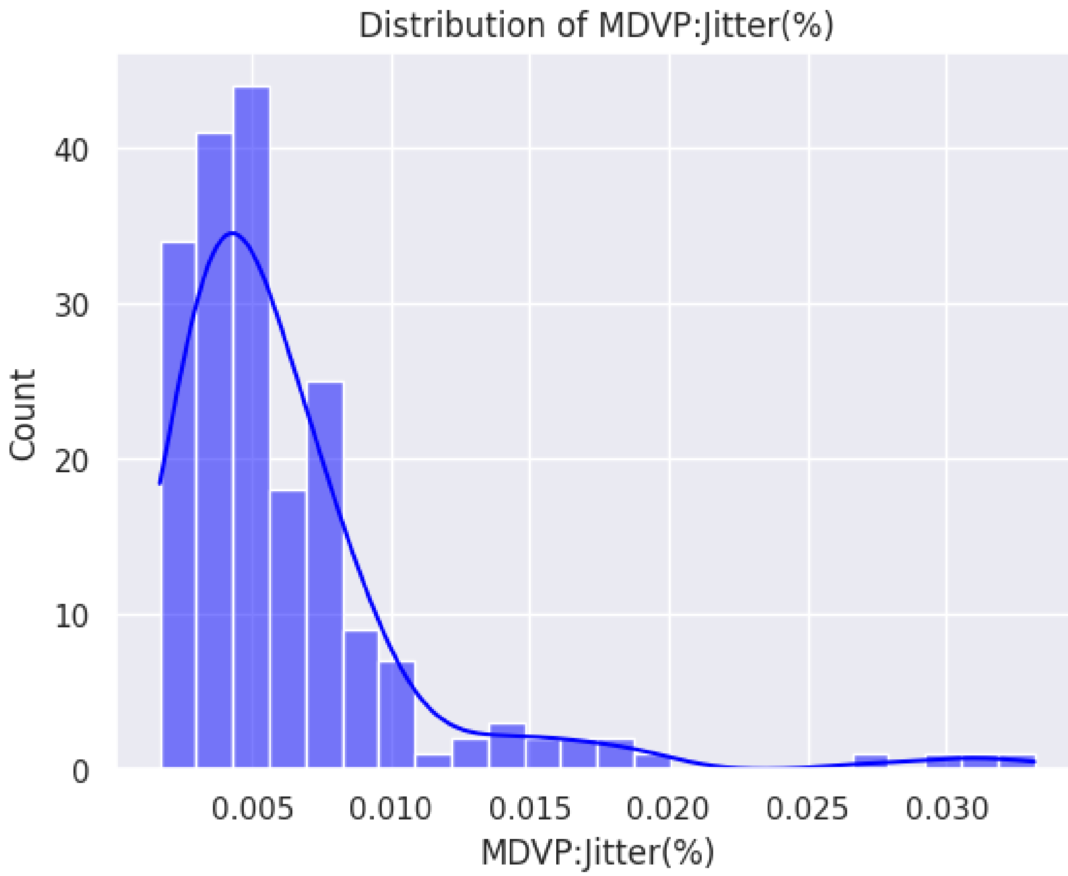
```
    plt.title(f"Distribution of {col}")
    plt.show()
```

```
Distribution plots for numeric features:
```

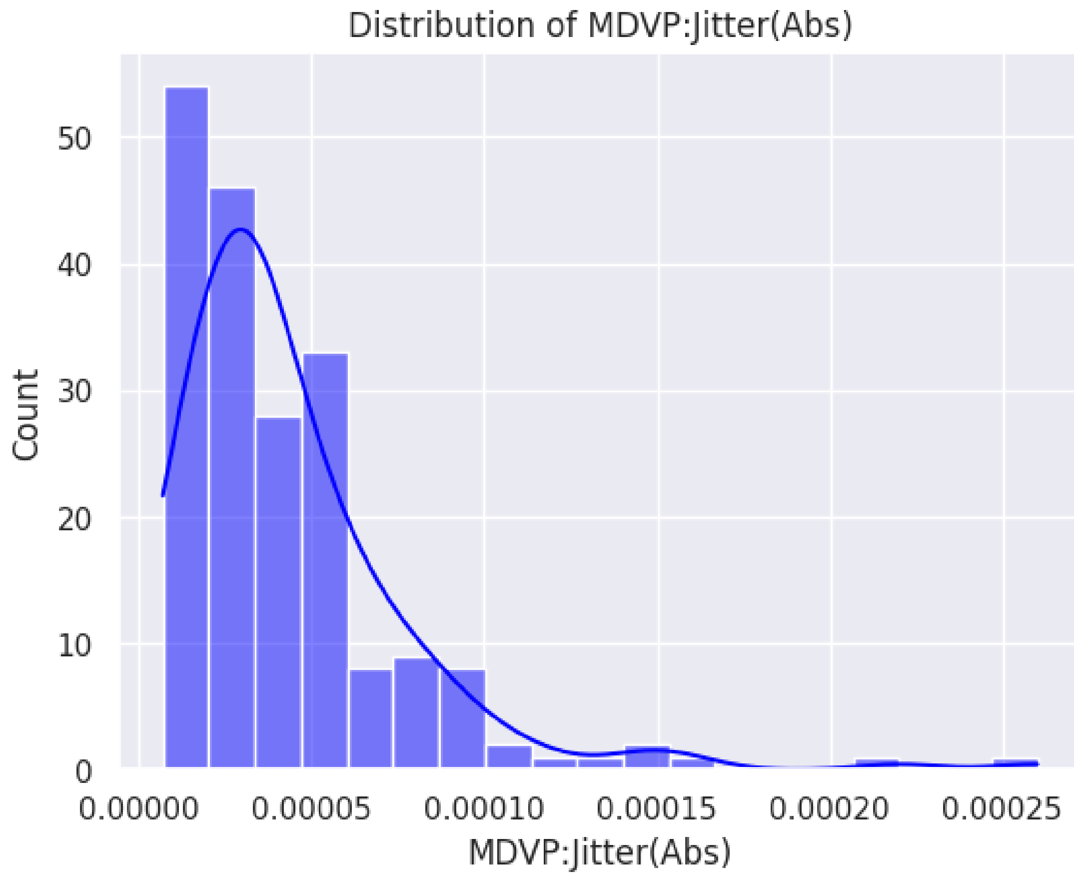



Distribution of MDVP:Fhi(Hz)
Count
0
10
20
30
40
100
200
300
400
500
600
MDVP:Fhi(Hz)

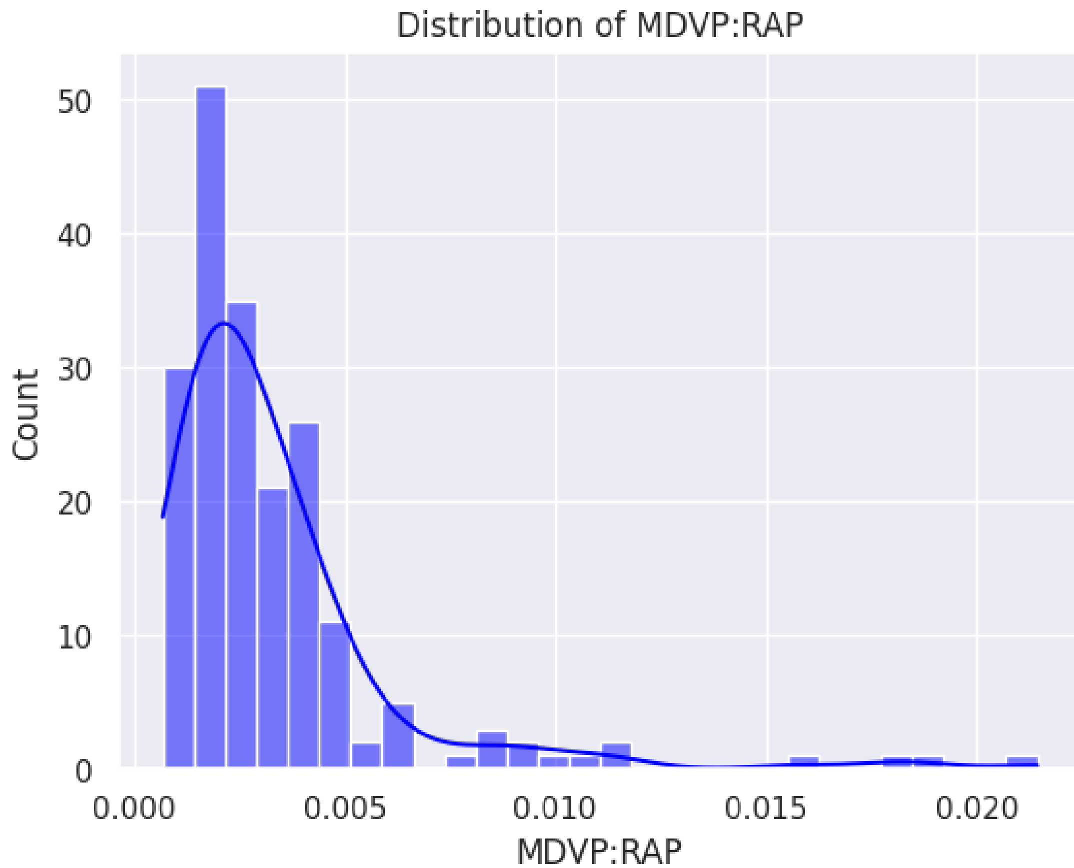

Distribution of MDVP:Flo(Hz)
Count
0
10
20
30
40
75
100
125
150
175
200
225
MDVP:Flo(Hz)



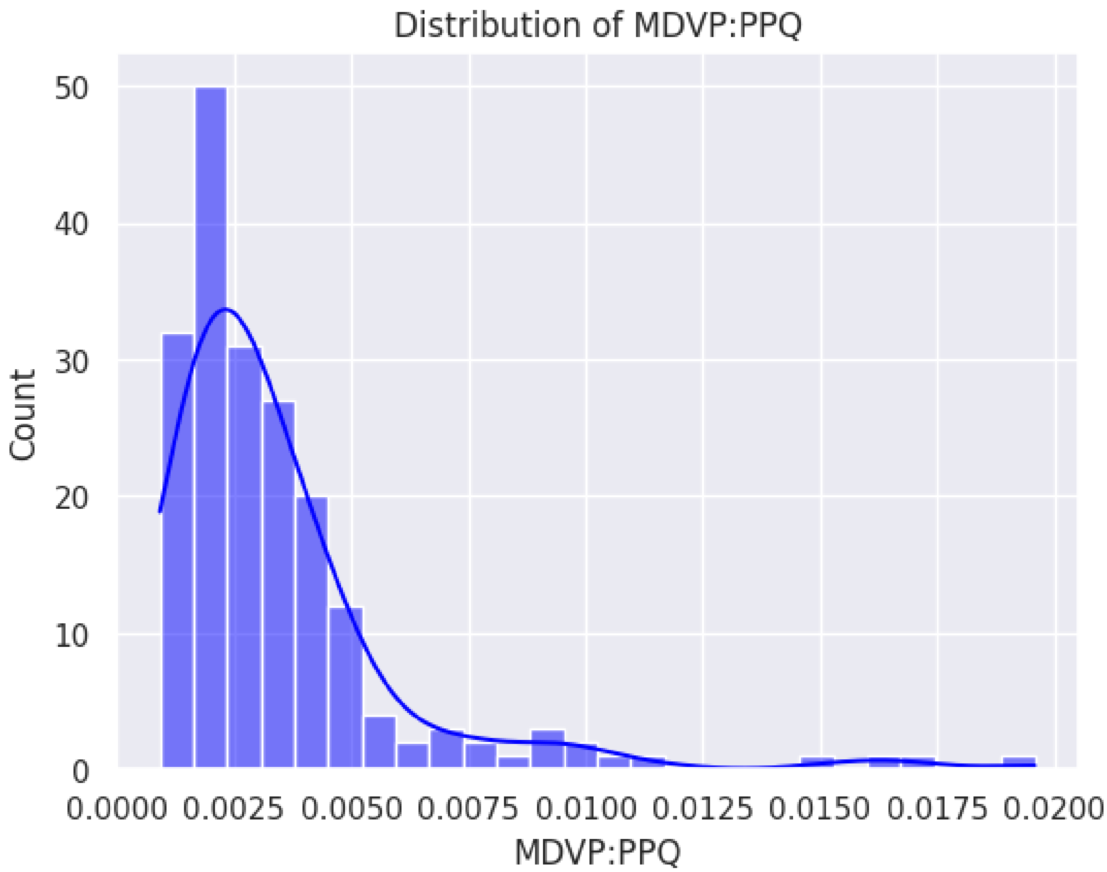
Distribution of MDVP:Jitter(%)
Count
MDVP:Jitter(%)
0
10
20
30
40
0.005
0.010
0.015
0.020
0.025
0.030



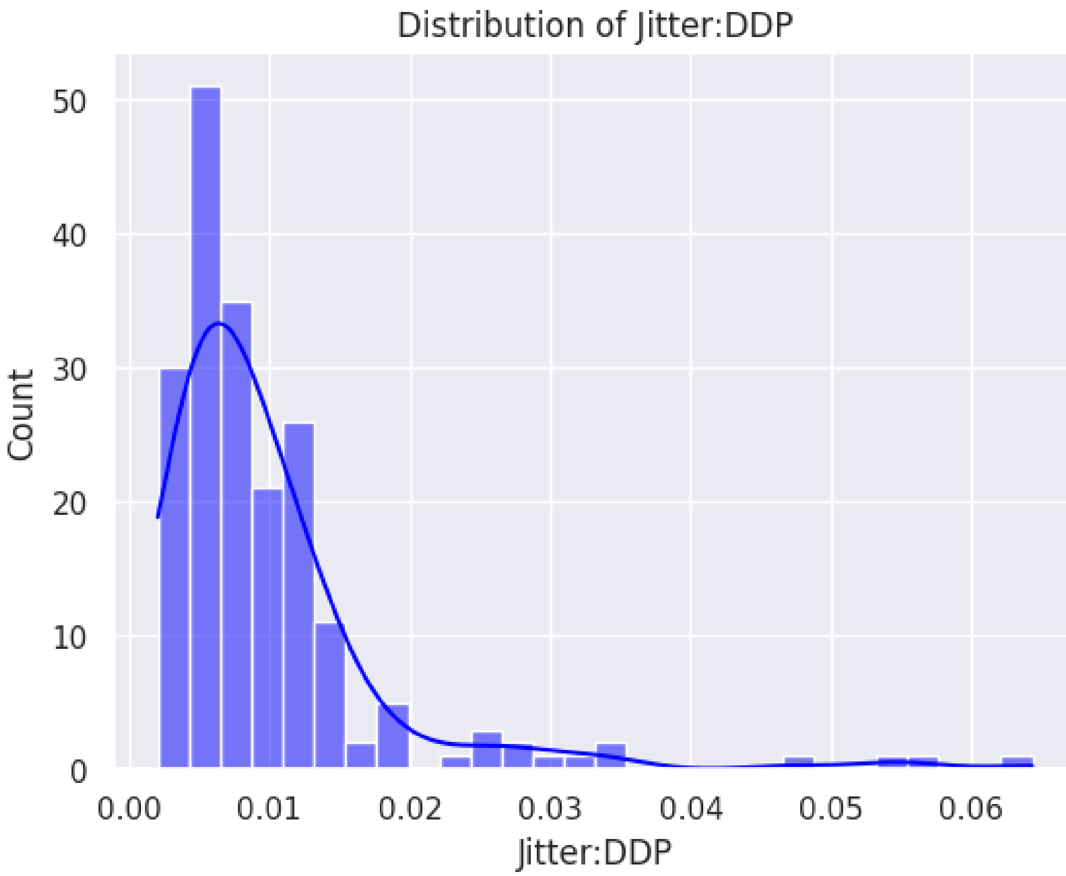
Distribution of MDVP:Jitter(Abs)
Count
0
10
20
30
40
50
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
MDVP:Jitter(Abs)



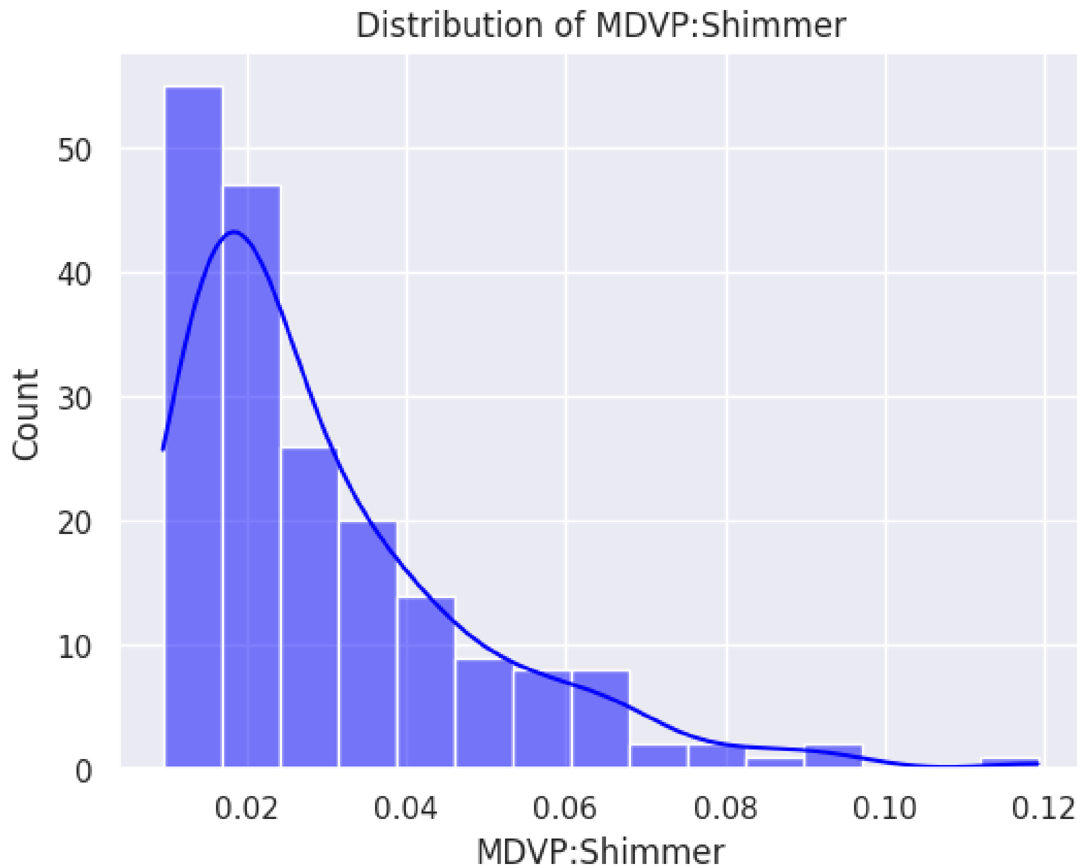
Distribution of MDVP:RAP
Count
50
40
30
20
10
0
0.000
0.005
0.010
0.015
0.020
MDVP:RAP



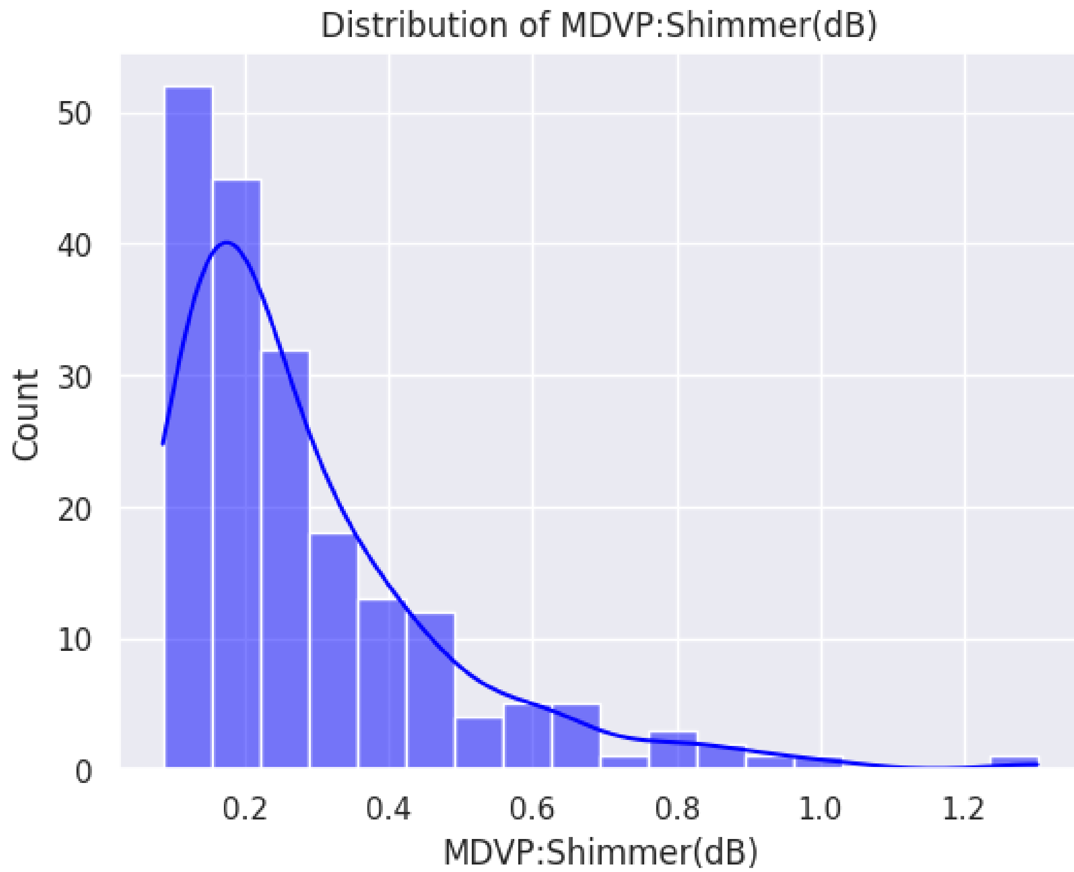
Distribution of MDVP:PPQ
Count
MDVP:PPQ
0
10
20
30
40
50
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200

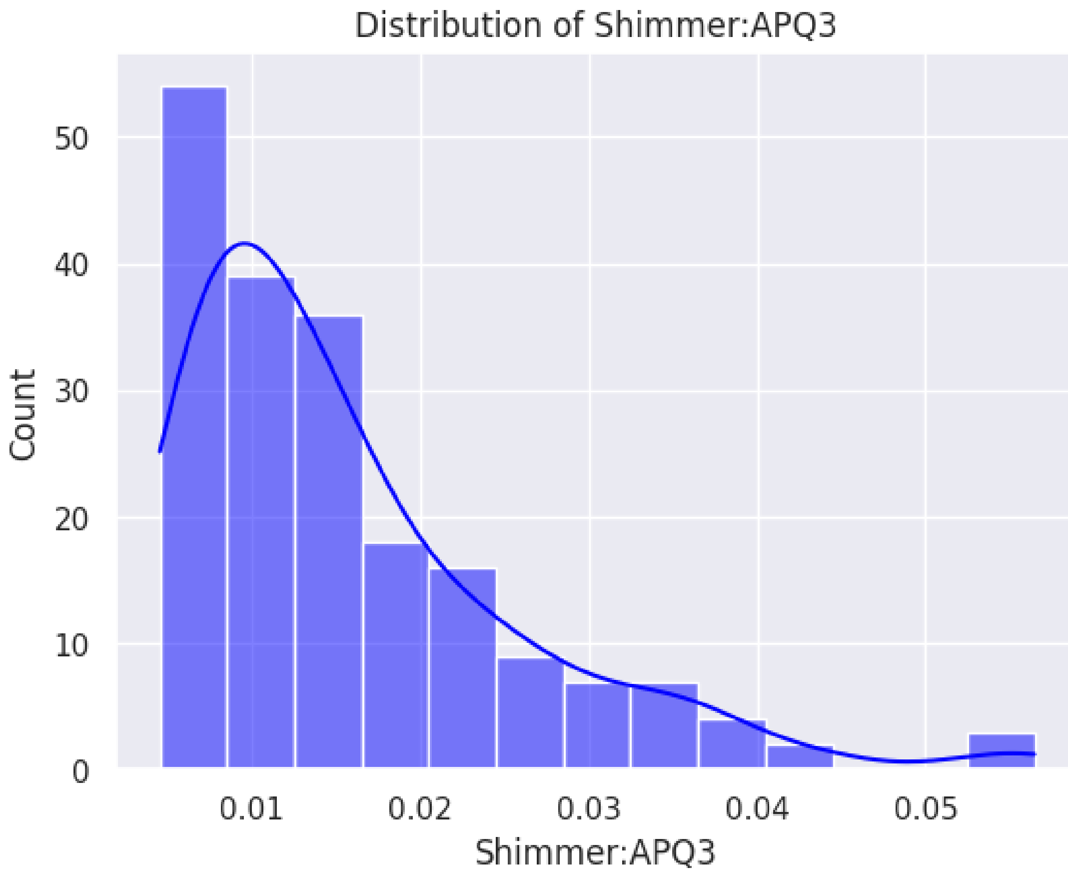


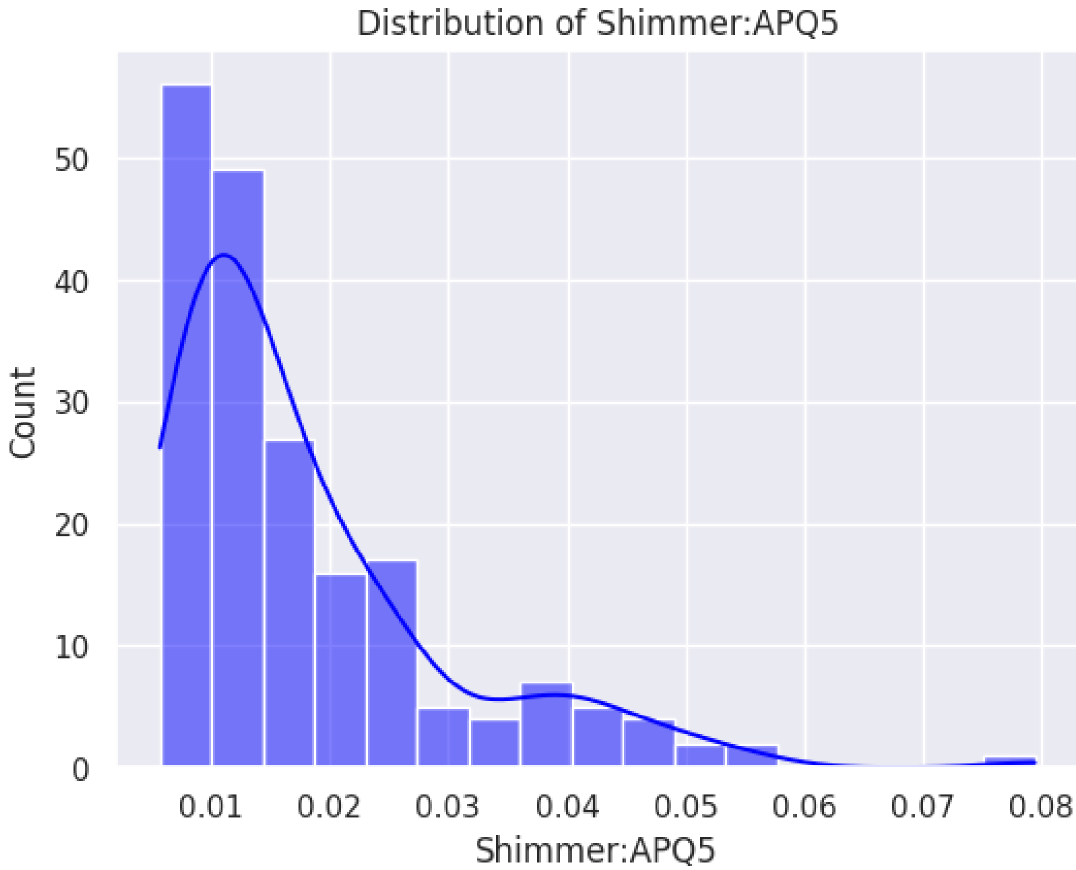
Distribution of Jitter:DDP
Count
Jitter:DDP
0
10
20
30
40
50
0.00
0.01
0.02
0.03
0.04
0.05
0.06

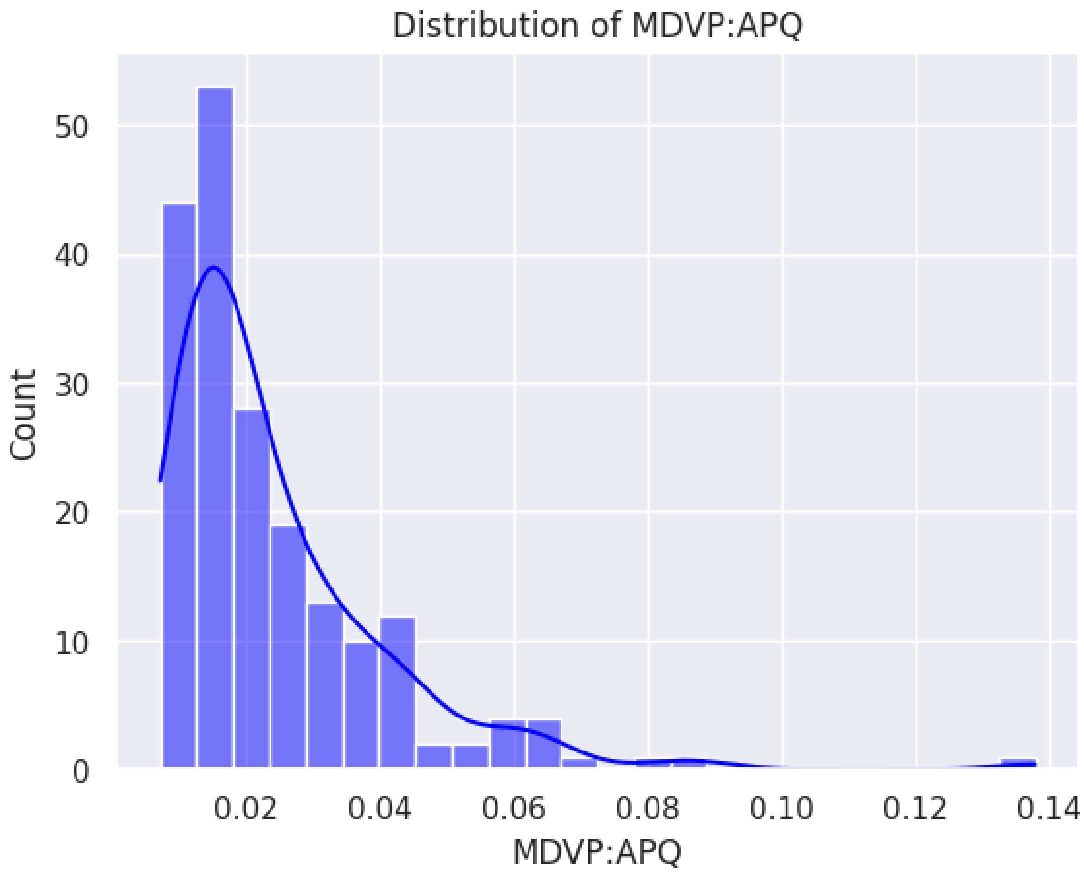


Distribution of MDVP:Shimmer
Count
0
10
20
30
40
50
0.02
0.04
0.06
0.08
0.10
0.12
MDVP:Shimmer

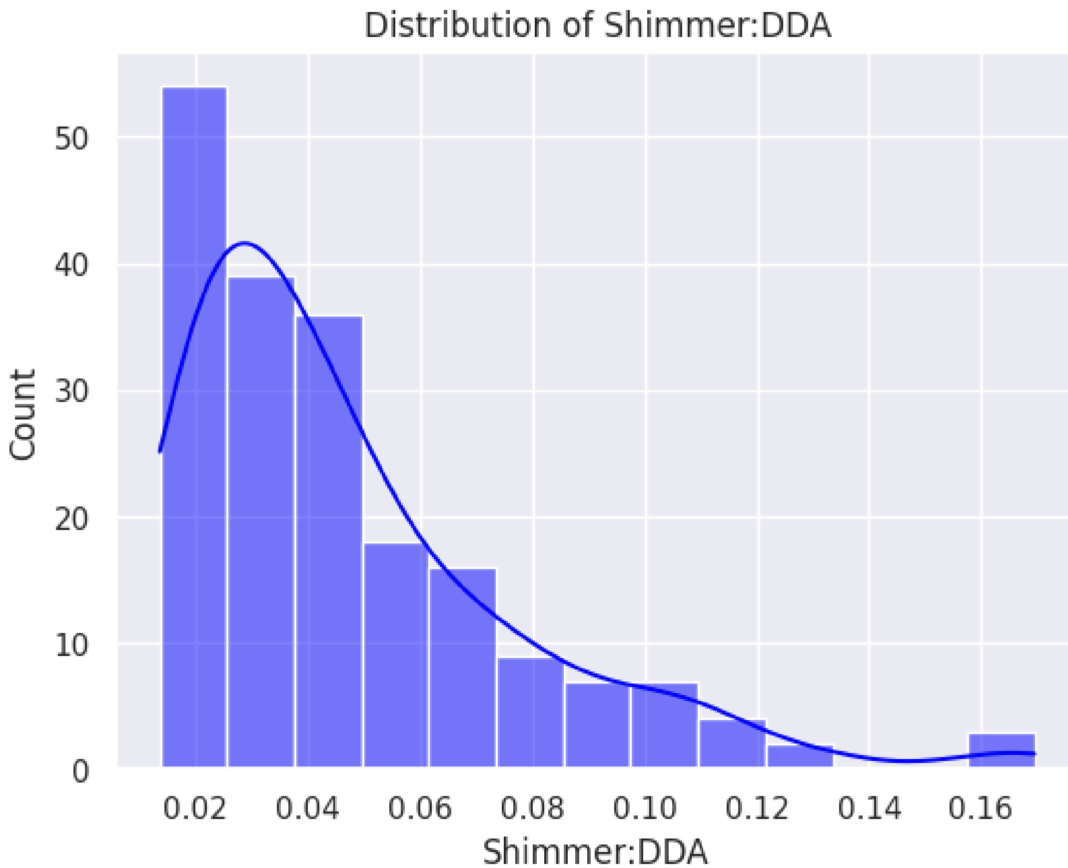

Distribution of MDVP:Shimmer(dB)
Count
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1.0
1.2
MDVP:Shimmer(dB)

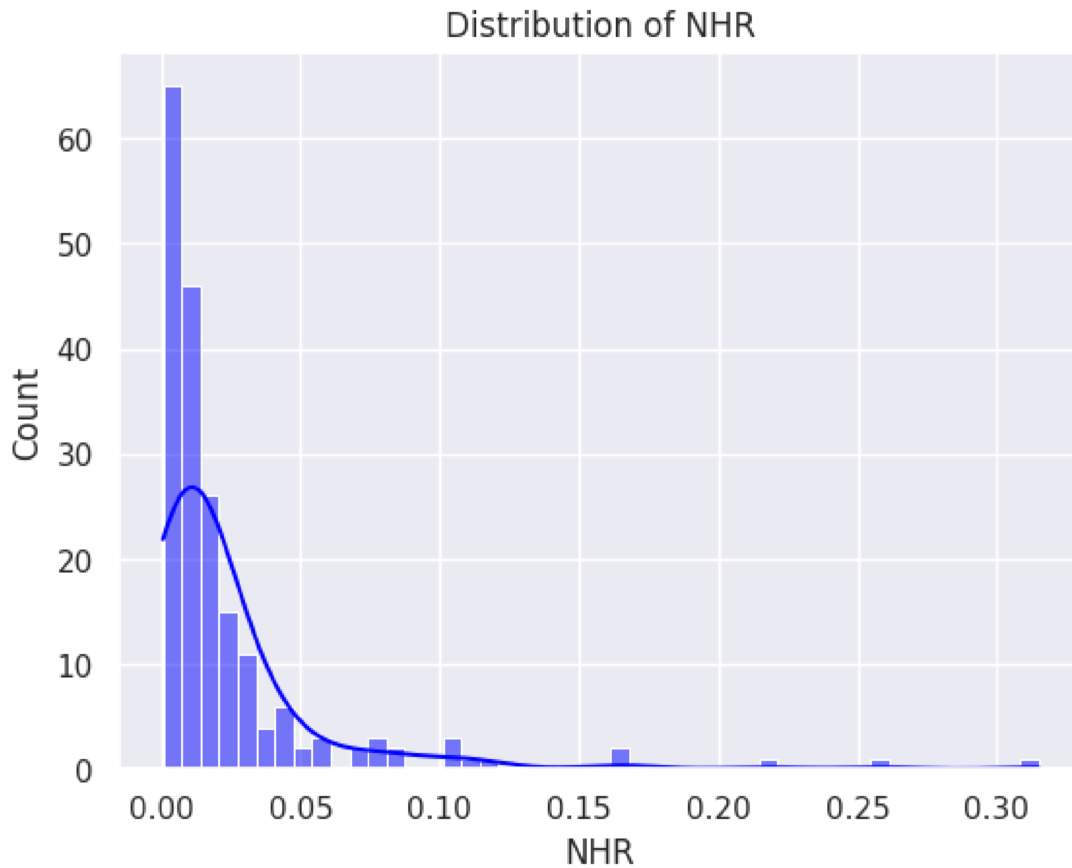

Distribution of Shimmer:APQ3
Count
0
10
20
30
40
50
0.01
0.02
0.03
0.04
0.05
Shimmer:APQ3

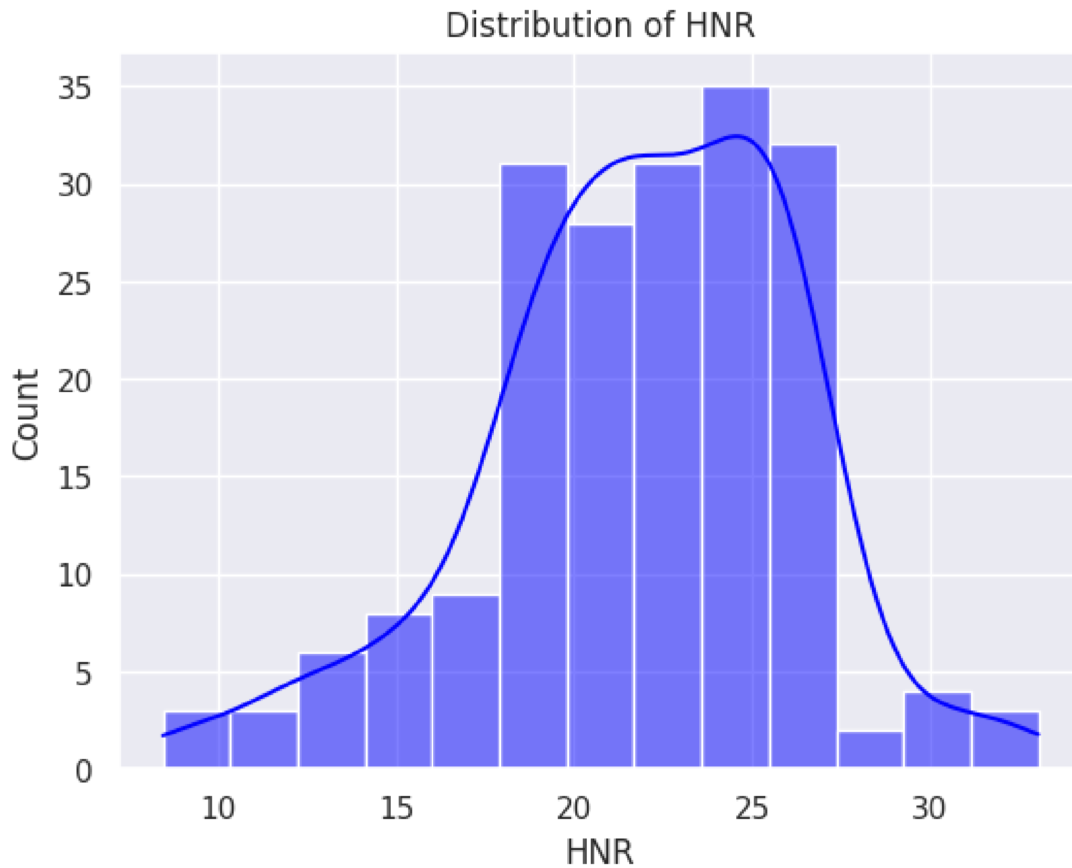

Distribution of Shimmer:APQ5
Count
50
40
30
20
10
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Shimmer:APQ5



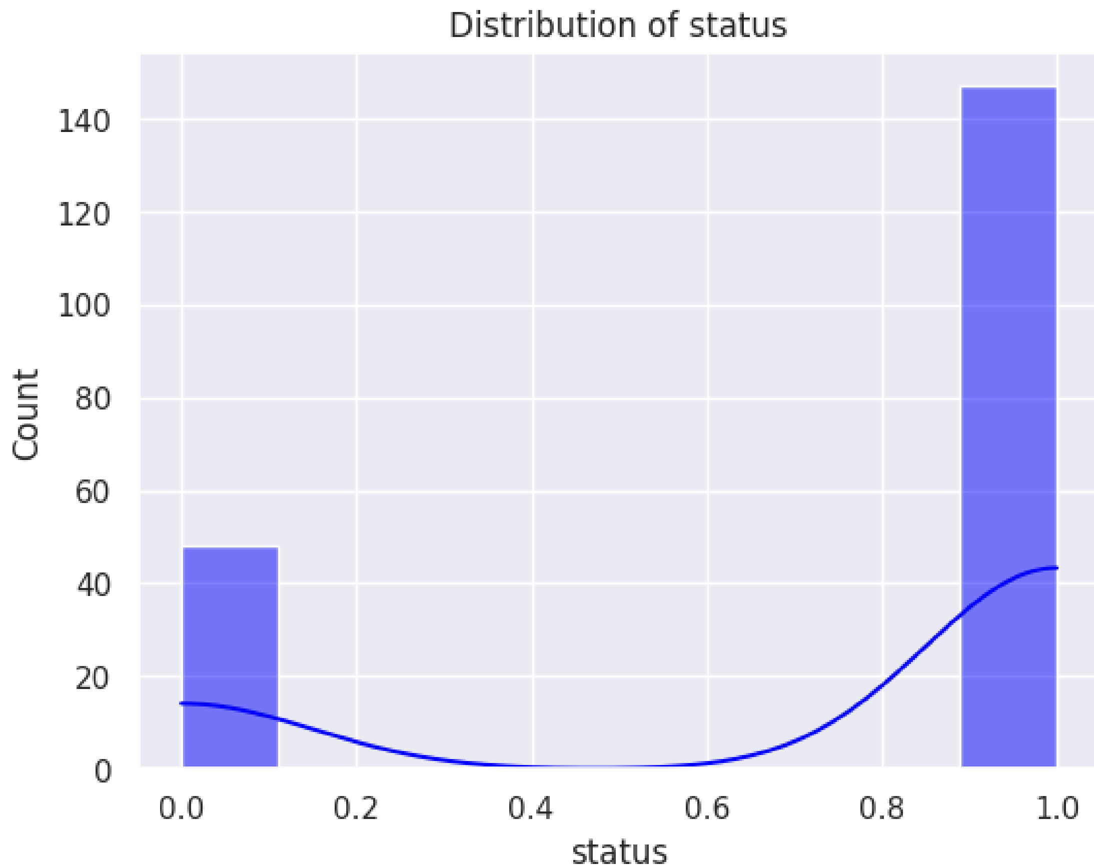
Distribution of MDVP:APQ
Count
MDVP:APQ
0
10
20
30
40
50
0.02
0.04
0.06
0.08
0.10
0.12
0.14

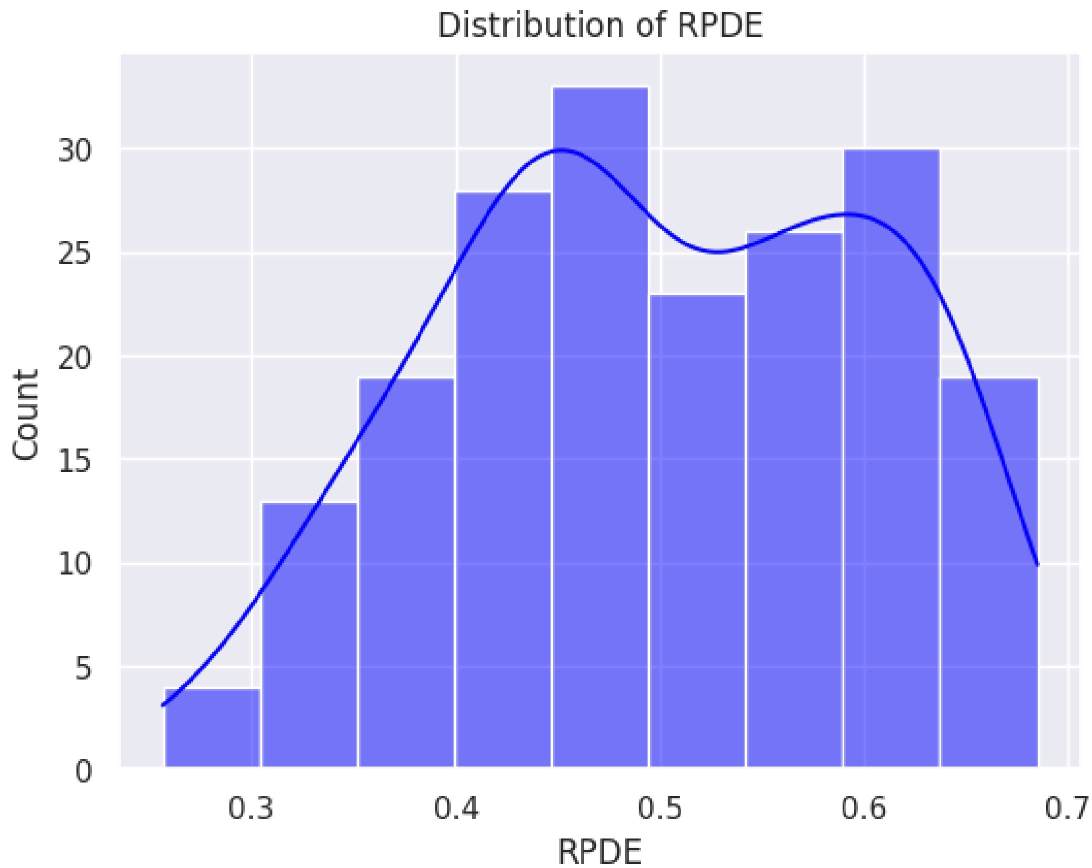

Distribution of Shimmer:DDA
Count
0
10
20
30
40
50
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Shimmer:DDA

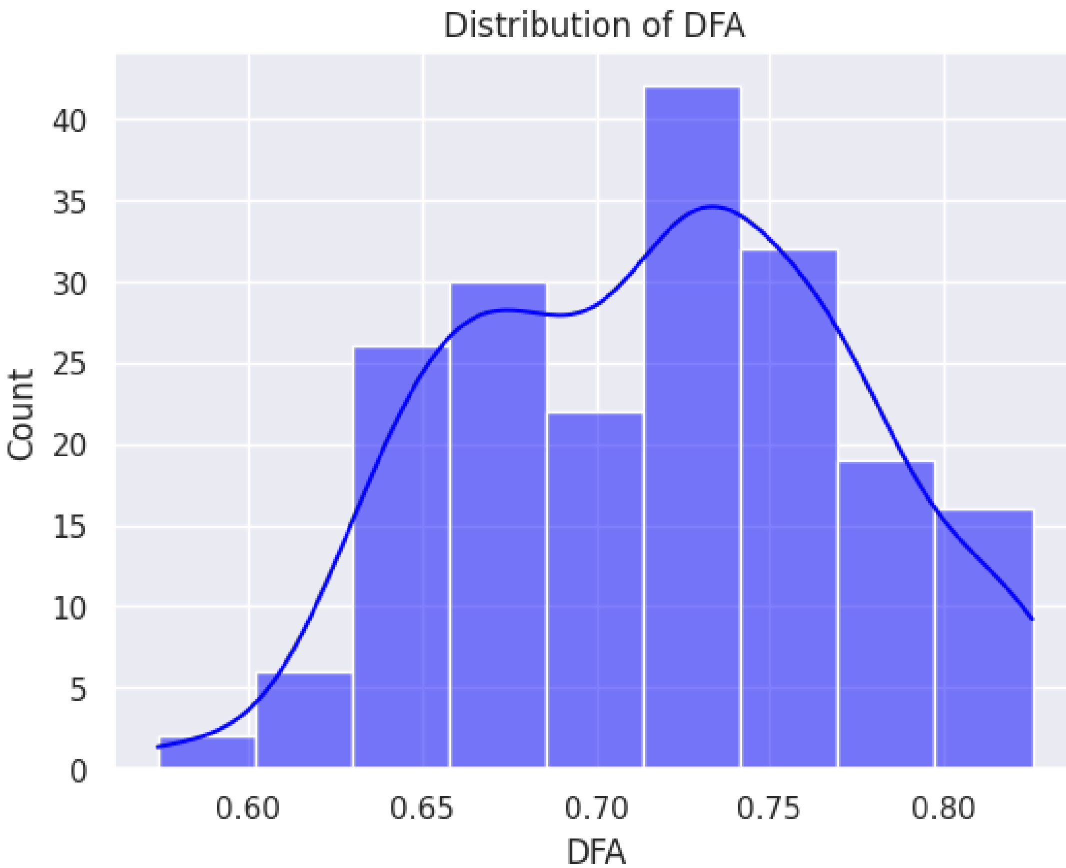

Distribution of NHR
Count
60
50
40
30
20
10
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
NHR

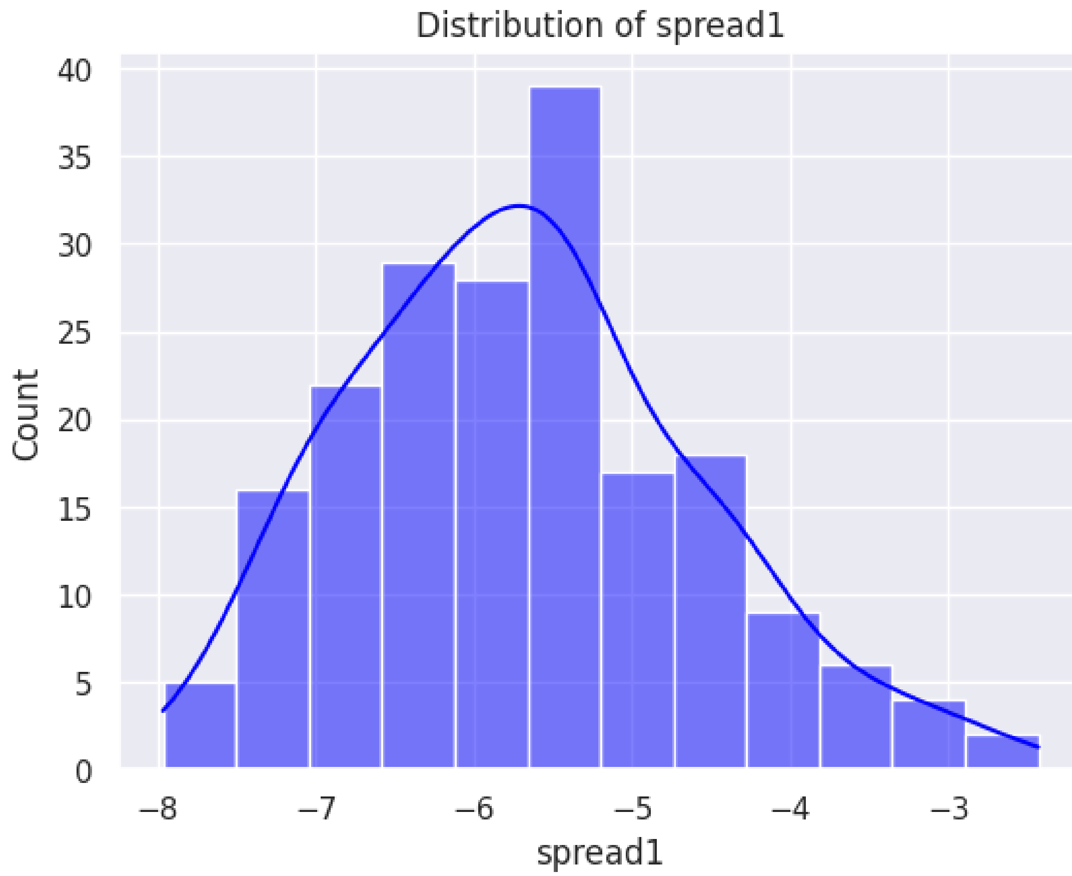


Distribution of HNR
Count
HNR
0
5
10
15
20
25
30
35
10
15
20
25
30

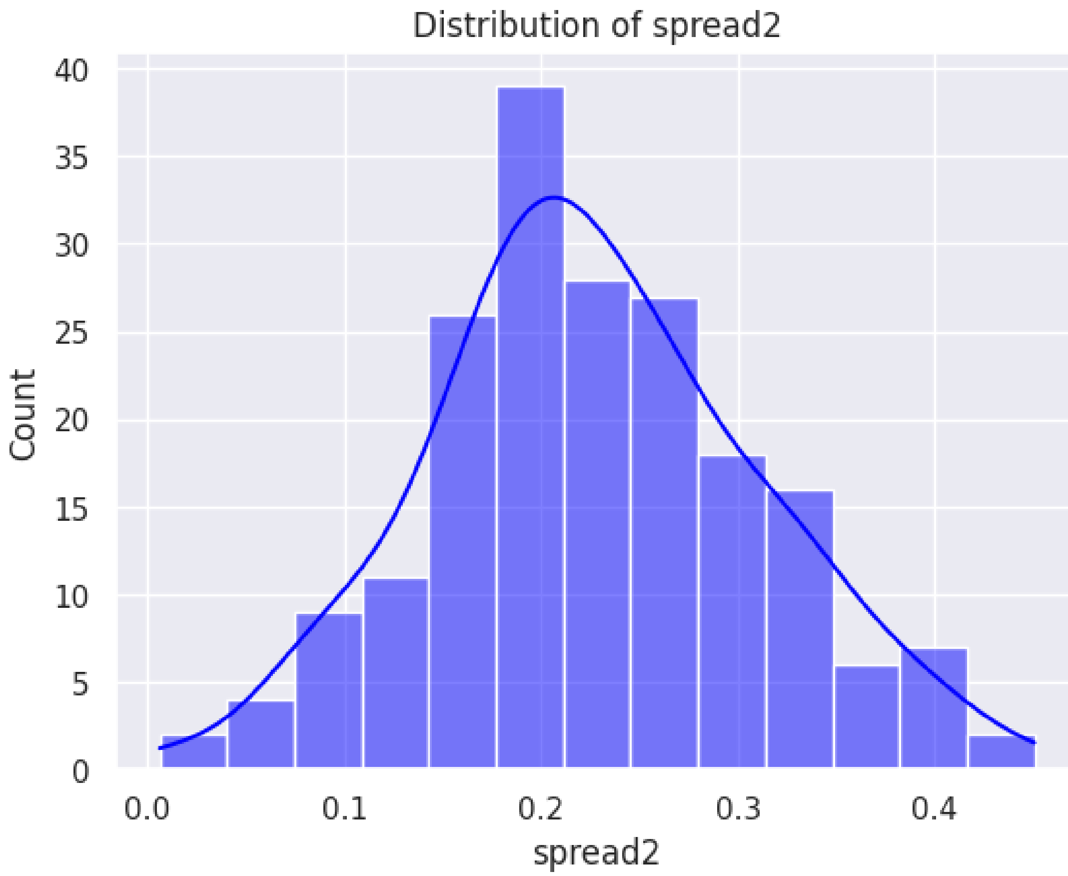

Distribution of status
Count
status
0
20
40
60
80
100
120
140
0.0
0.2
0.4
0.6
0.8
1.0

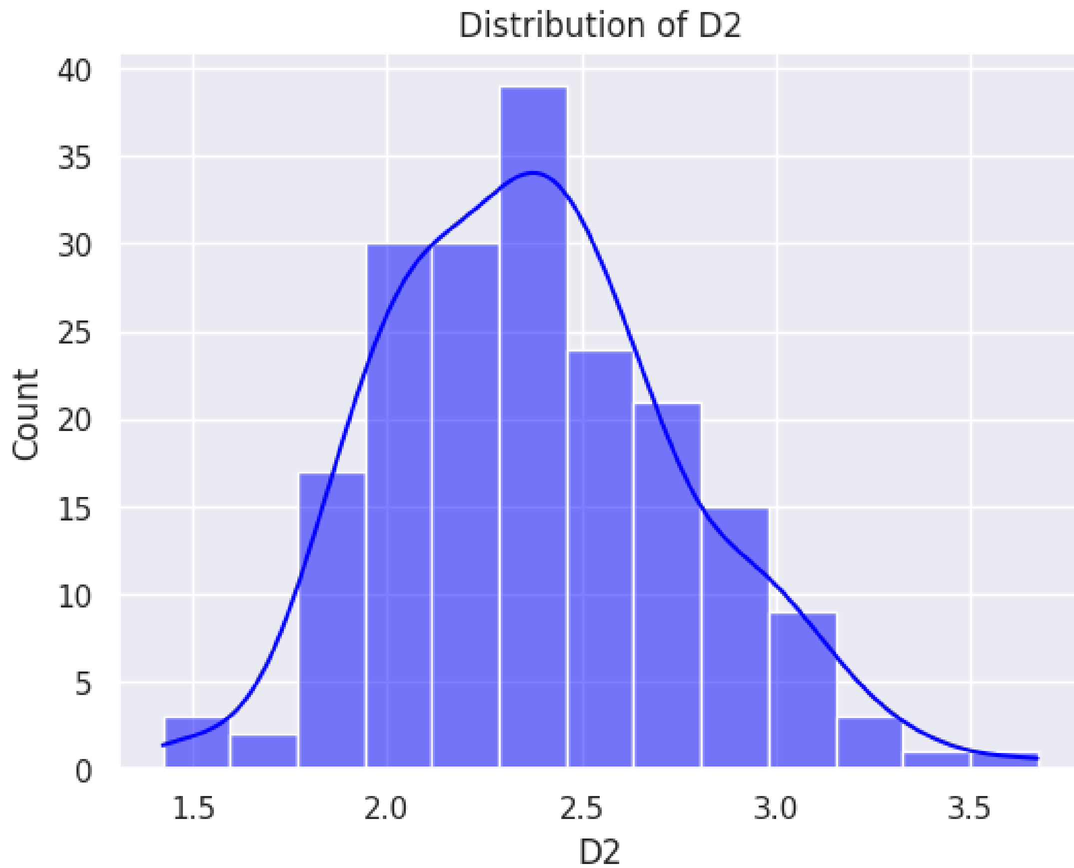

Distribution of RPDE
Count
RPDE
0
5
10
15
20
25
30
0.3
0.4
0.5
0.6
0.7

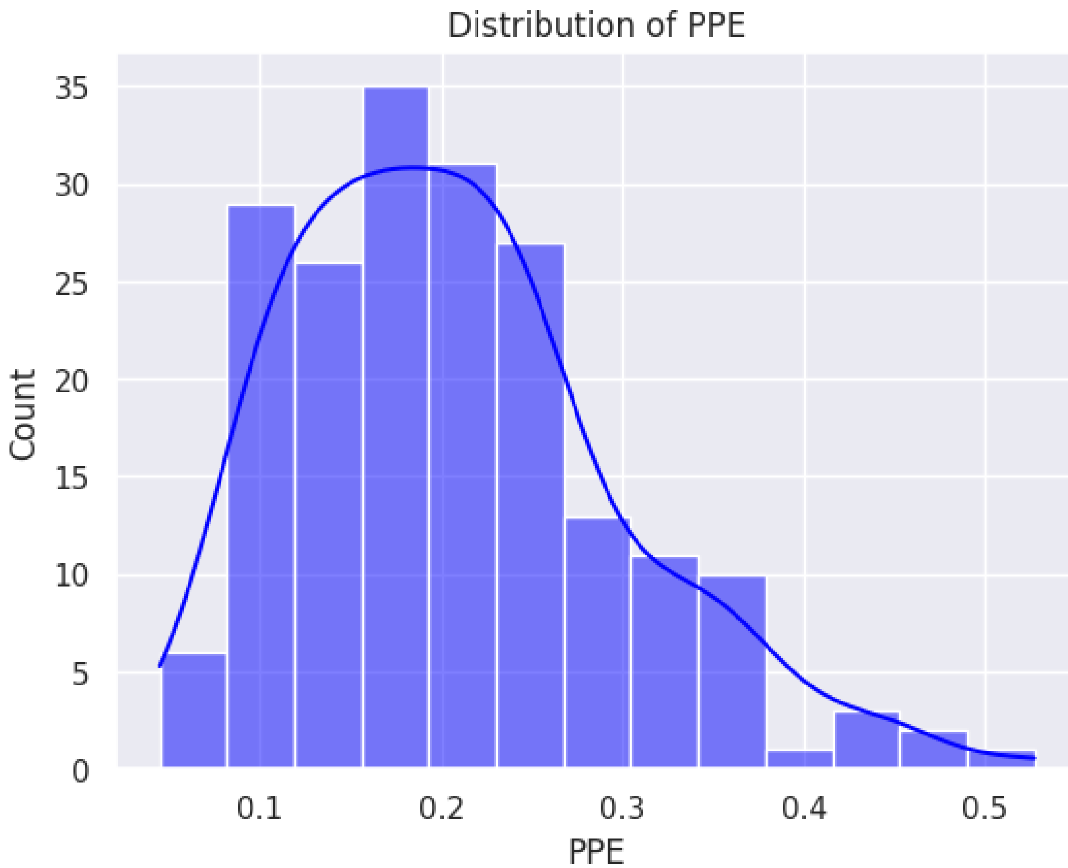

Distribution of DFA
Count
40
35
30
25
20
15
10
5
0
0.60
0.65
0.70
0.75
0.80
DFA



Distribution of spread1
Count
spread1
0
5
10
15
20
25
30
35
40
−8
−7
−6
−5
−4
−3

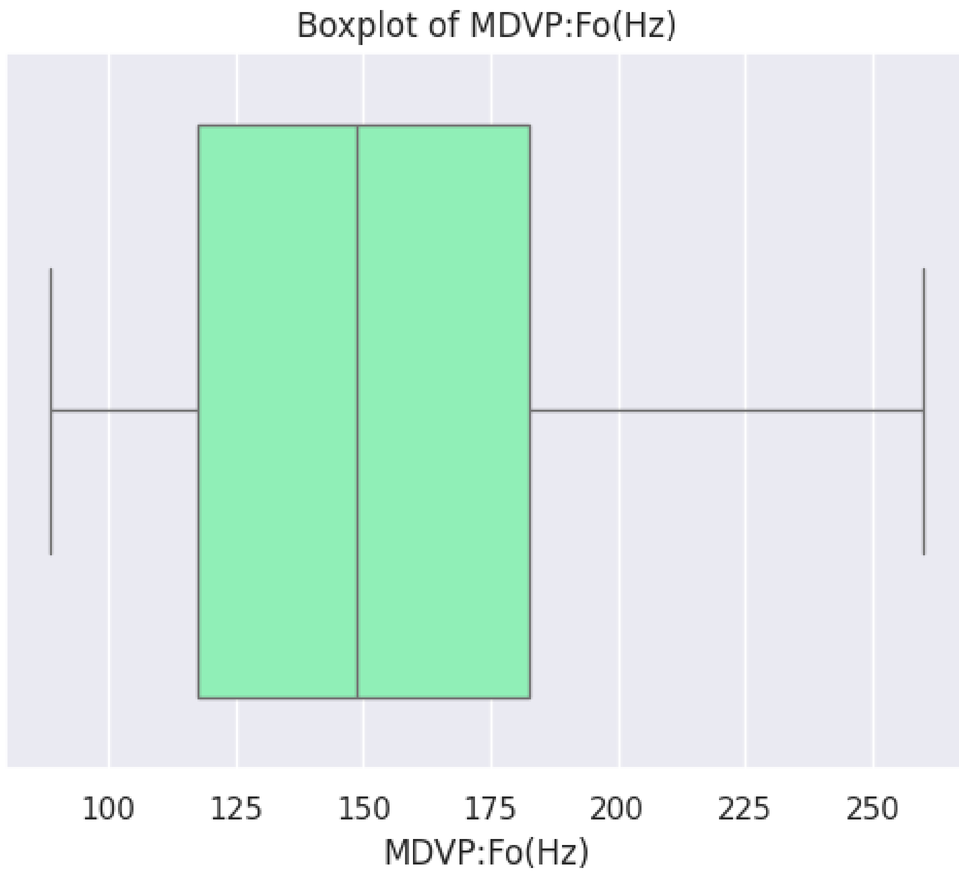

Distribution of spread2
Count
spread2
0
5
10
15
20
25
30
35
40
0.0
0.1
0.2
0.3
0.4

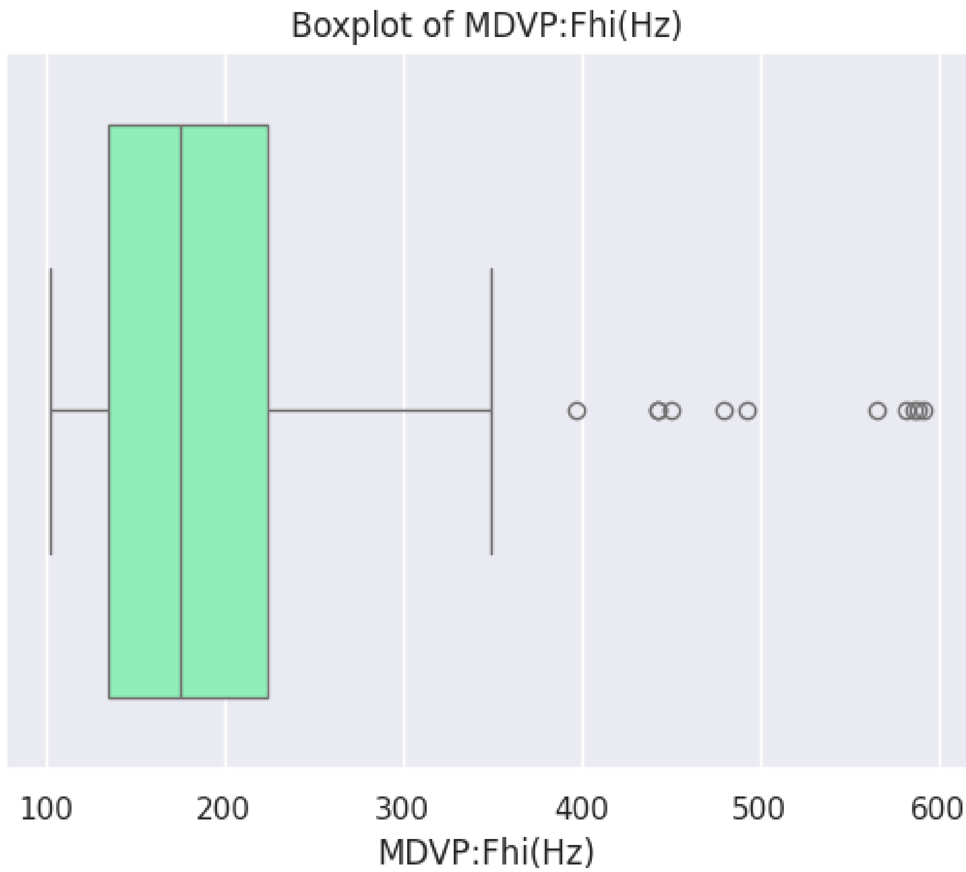

Distribution of D2
Count
D2
0
5
10
15
20
25
30
35
40
1.5
2.0
2.5
3.0
3.5

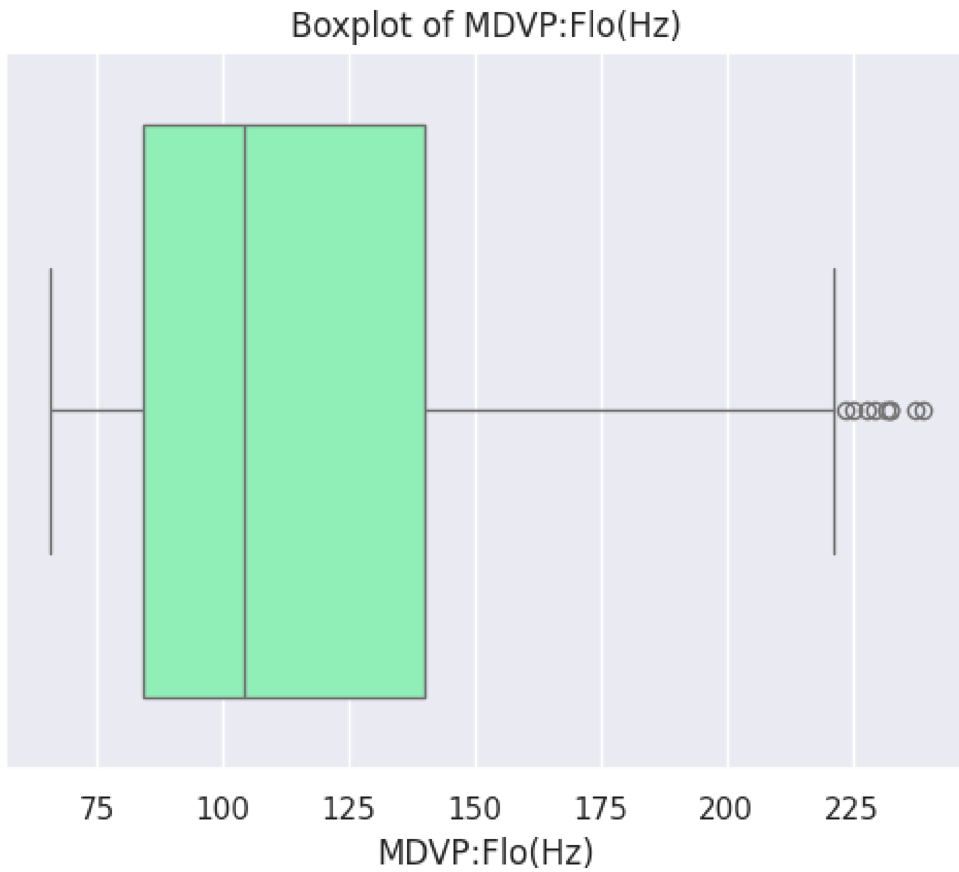


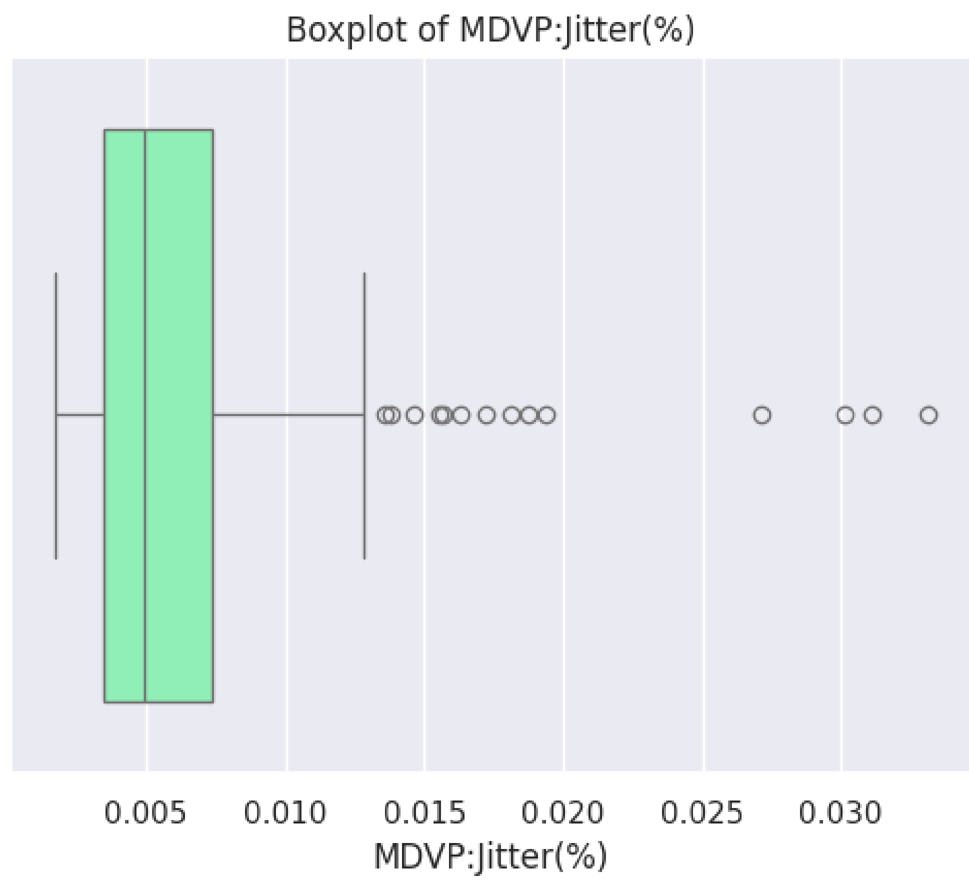
Checking whether there are any outliers

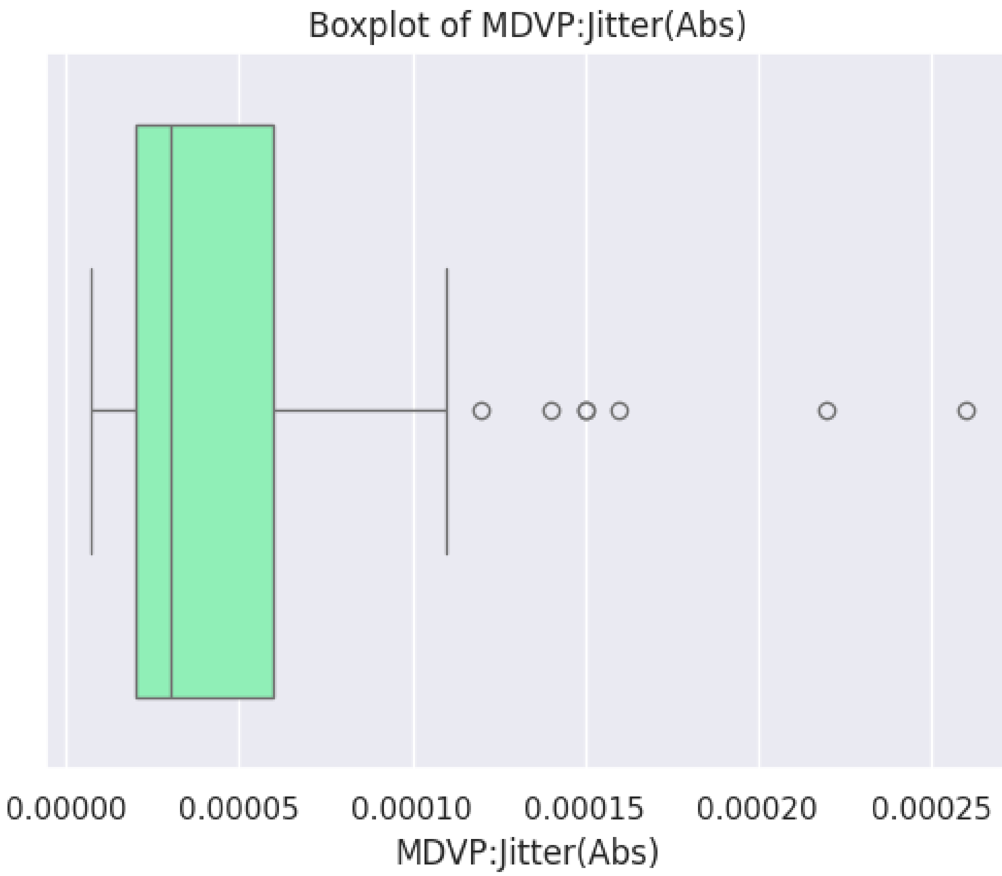
```
# Boxplots to check for outliers
print("\nBoxplots for numeric features to check for outliers:")
for col in df.select_dtypes(include=['number']).columns:
    sns.boxplot(x=df[col], palette='rainbow')
    plt.title(f"Boxplot of {col}")
    plt.show()


Boxplots for numeric features to check for outliers:
```

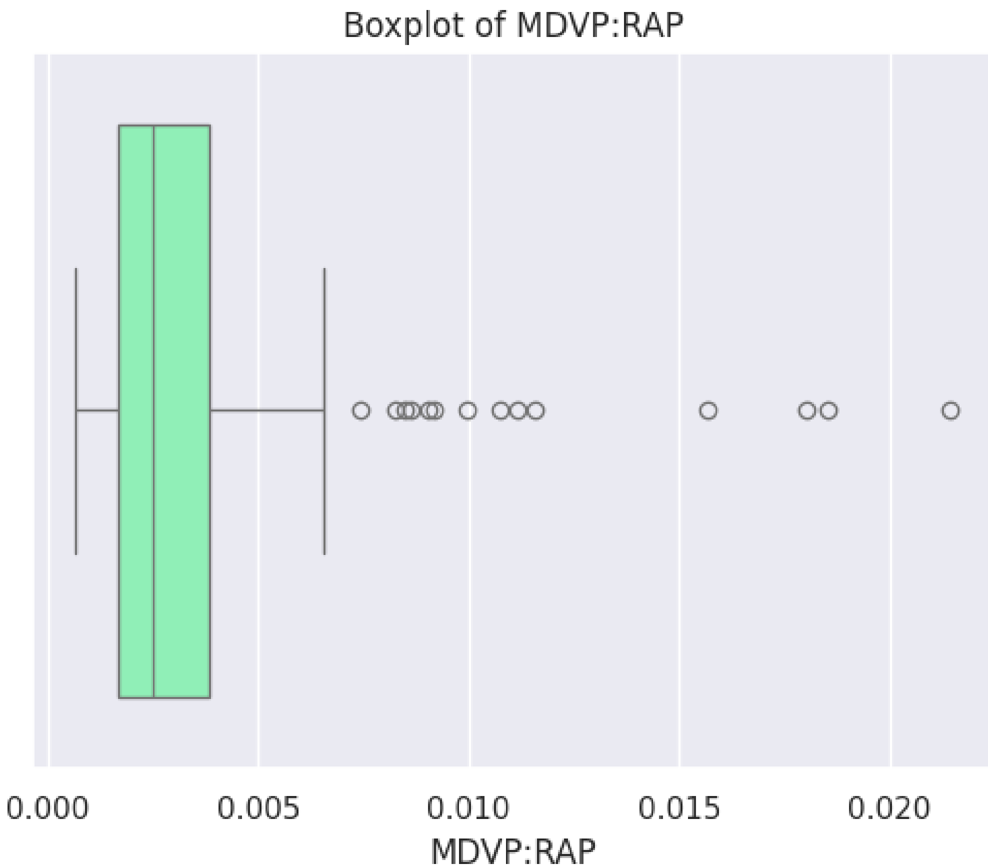

Boxplot of MDVP:Fo(Hz)
100
125
150
175
200
225
250
MDVP:Fo(Hz)

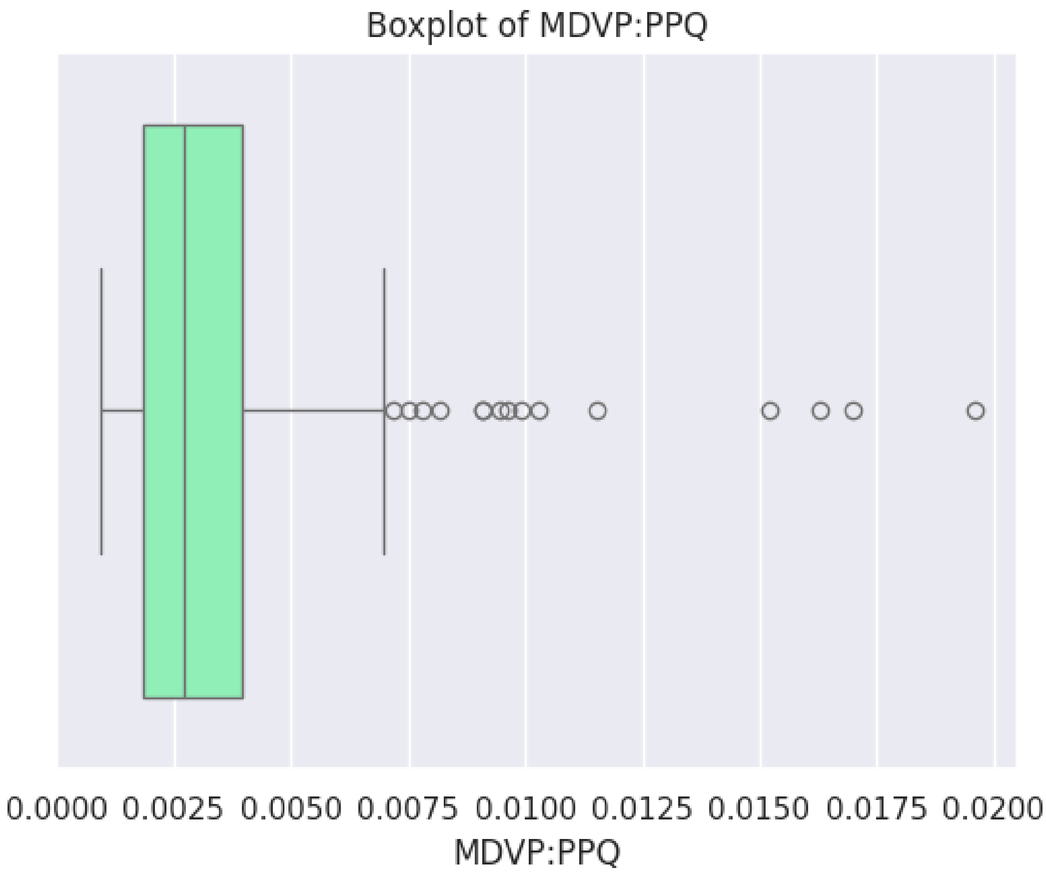

Boxplot of MDVP:Fhi(Hz)
100
200
300
400
500
600
MDVP:Fhi(Hz)


Boxplot of MDVP:Flo(Hz)
75
100
125
150
175
200
225
MDVP:Flo(Hz)

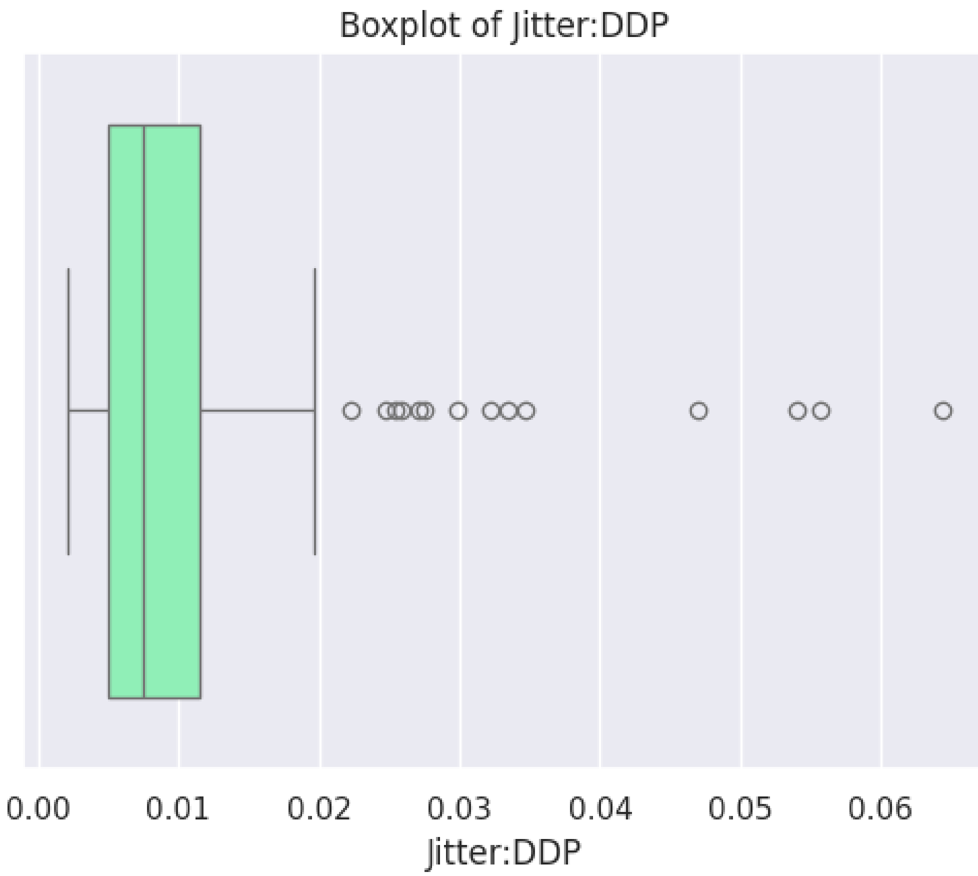

Boxplot of MDVP:Jitter(%)
0.005
0.010
0.015
0.020
0.025
0.030
MDVP:Jitter(%)


Boxplot of MDVP:Jitter(Abs)
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
MDVP:Jitter(Abs)

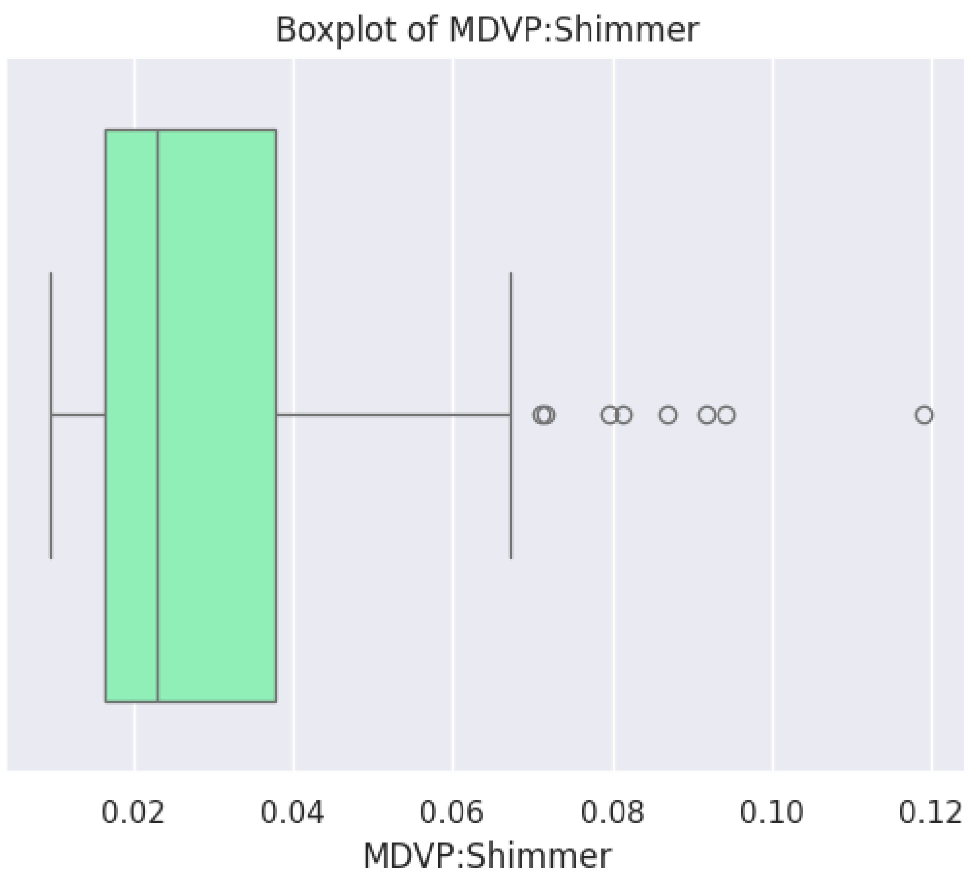

Boxplot of MDVP:RAP
0.000
0.005
0.010
0.015
0.020
MDVP:RAP

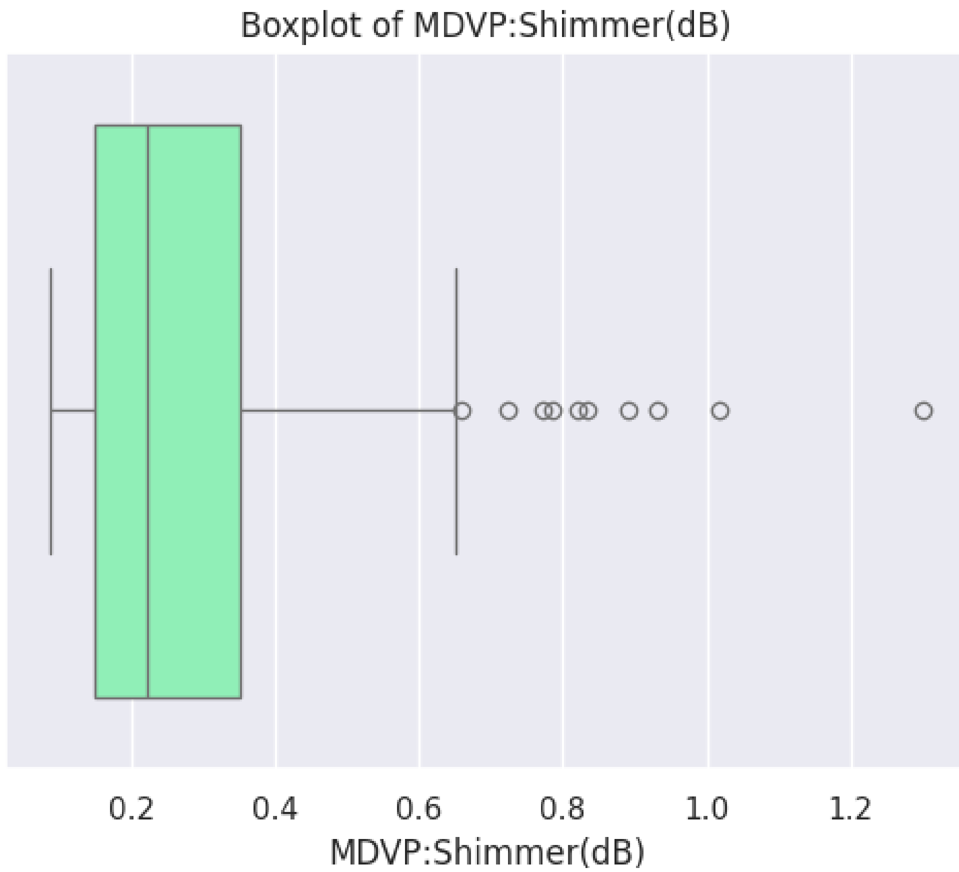


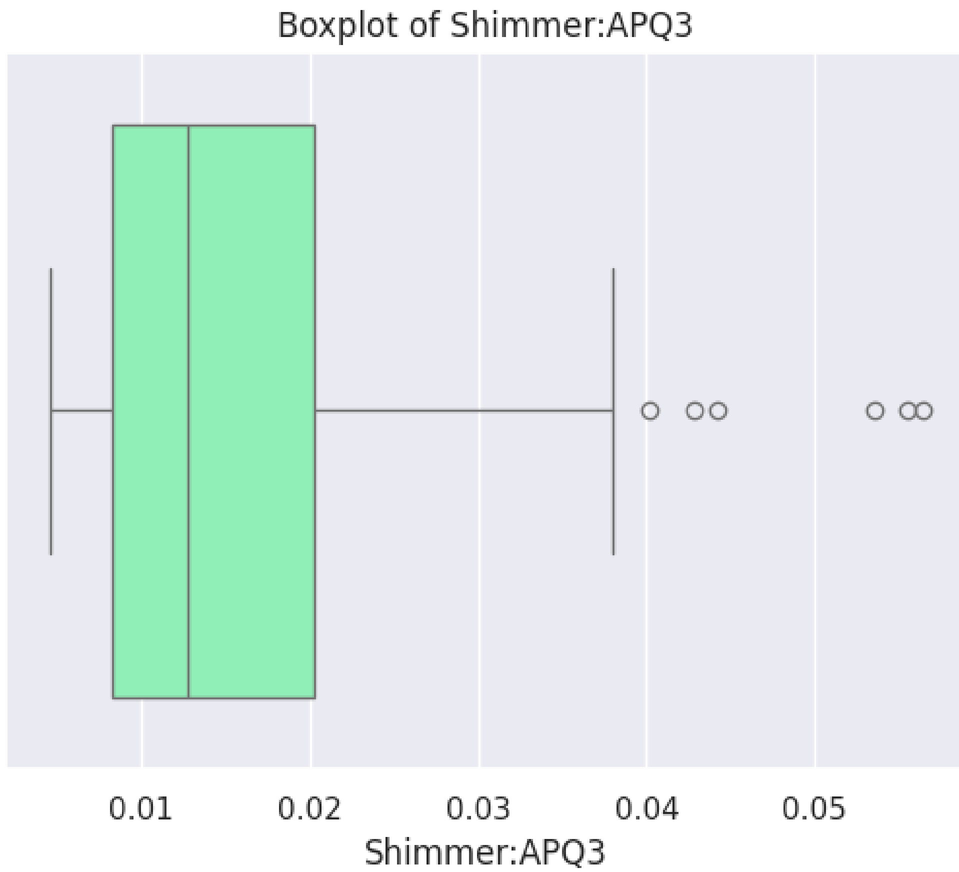
Boxplot of MDVP:PPQ
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
MDVP:PPQ

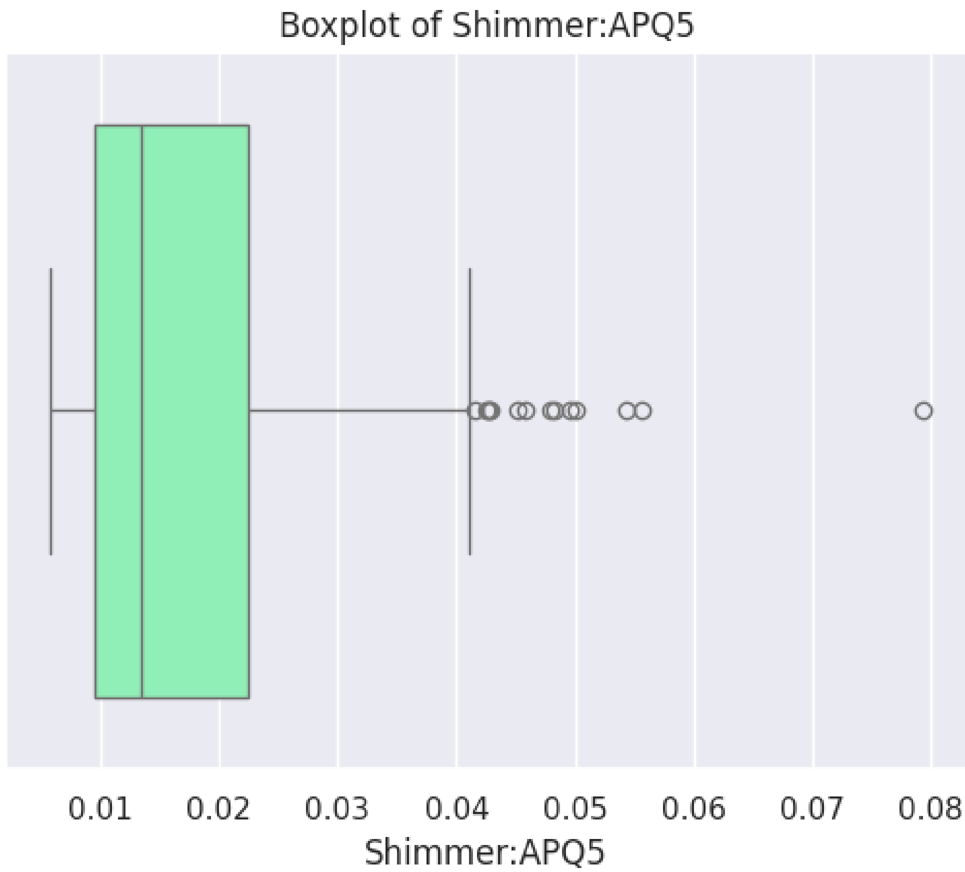


Boxplot of Jitter:DDP
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Jitter:DDP

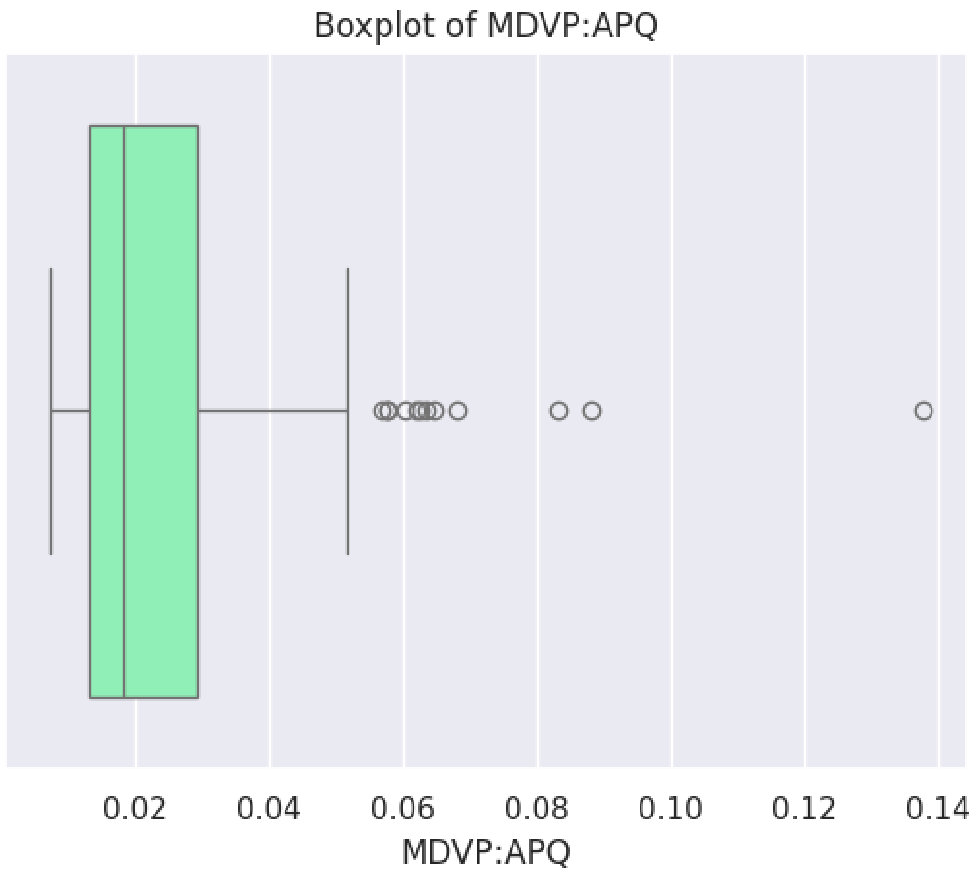

Boxplot of MDVP:Shimmer
0.02
0.04
0.06
0.08
0.10
0.12
MDVP:Shimmer

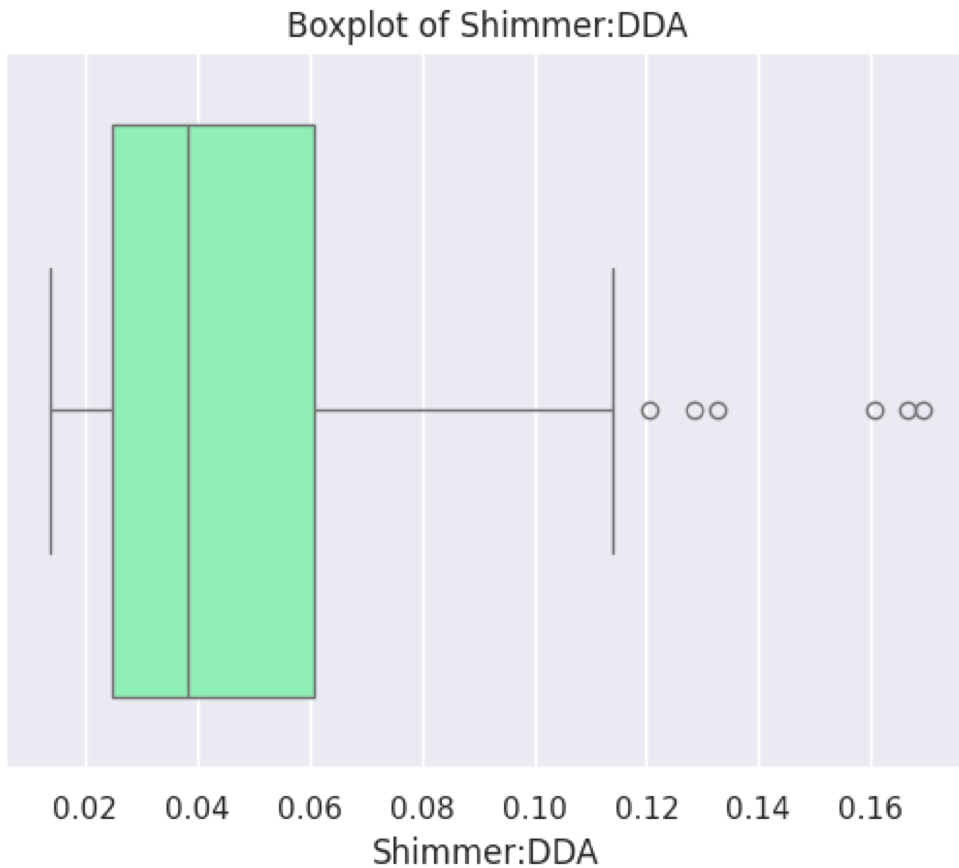

Boxplot of MDVP:Shimmer(dB)
0.2
0.4
0.6
0.8
1.0
1.2
MDVP:Shimmer(dB)

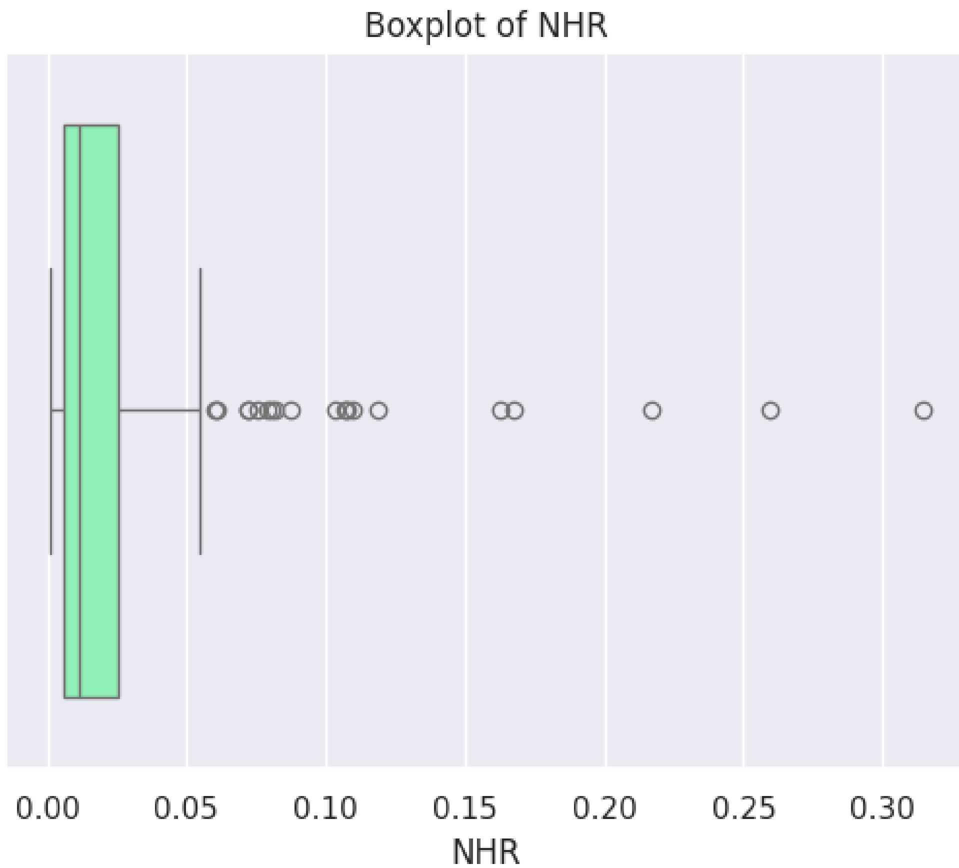

Boxplot of Shimmer:APQ3
0.01
0.02
0.03
0.04
0.05
Shimmer:APQ3



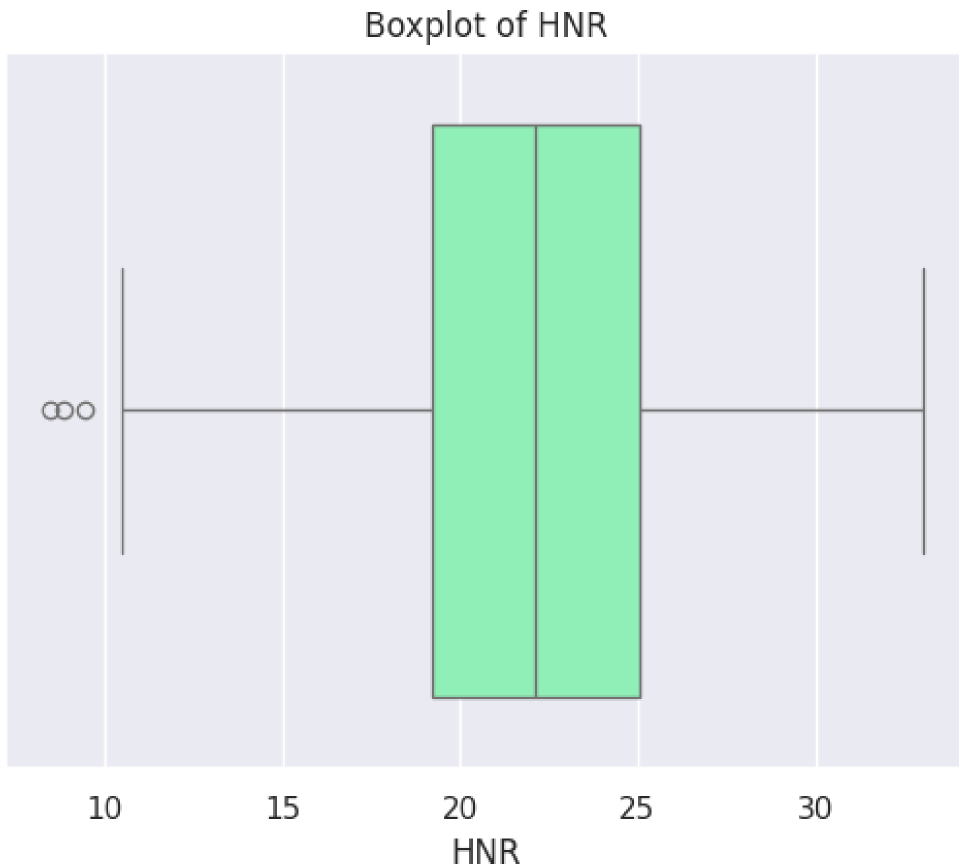
Boxplot of Shimmer:APQ5
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Shimmer:APQ5

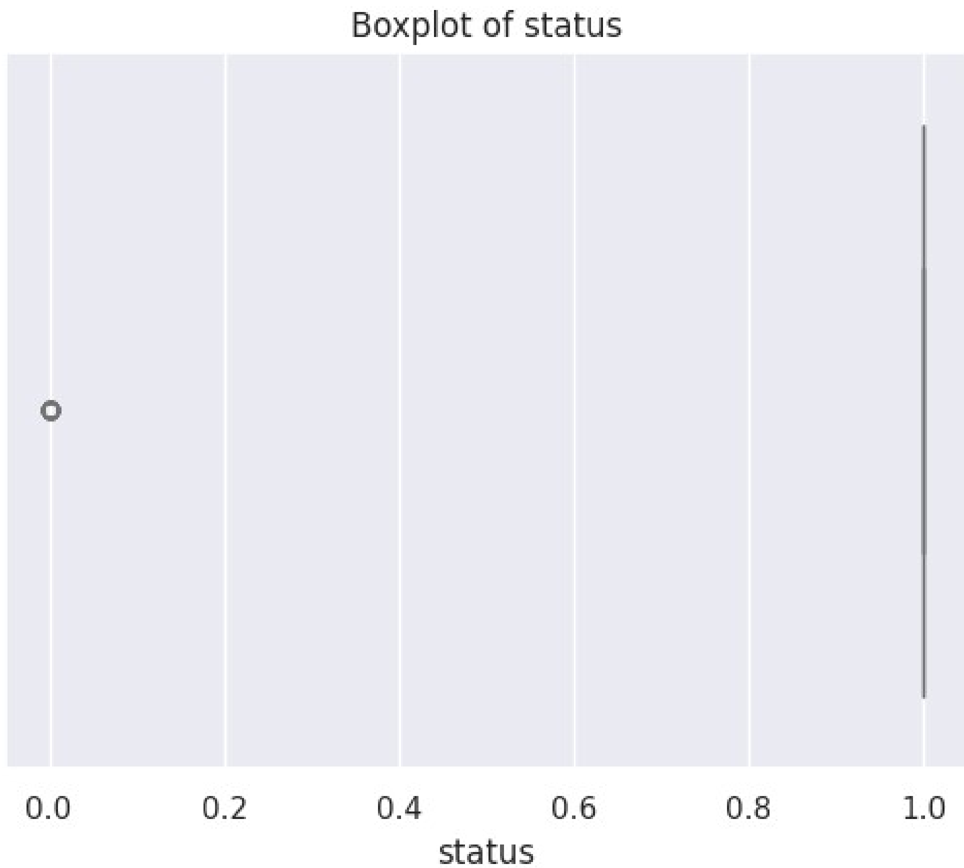

Boxplot of MDVP:APQ
0.02
0.04
0.06
0.08
0.10
0.12
0.14
MDVP:APQ


Boxplot of Shimmer:DDA
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Shimmer:DDA

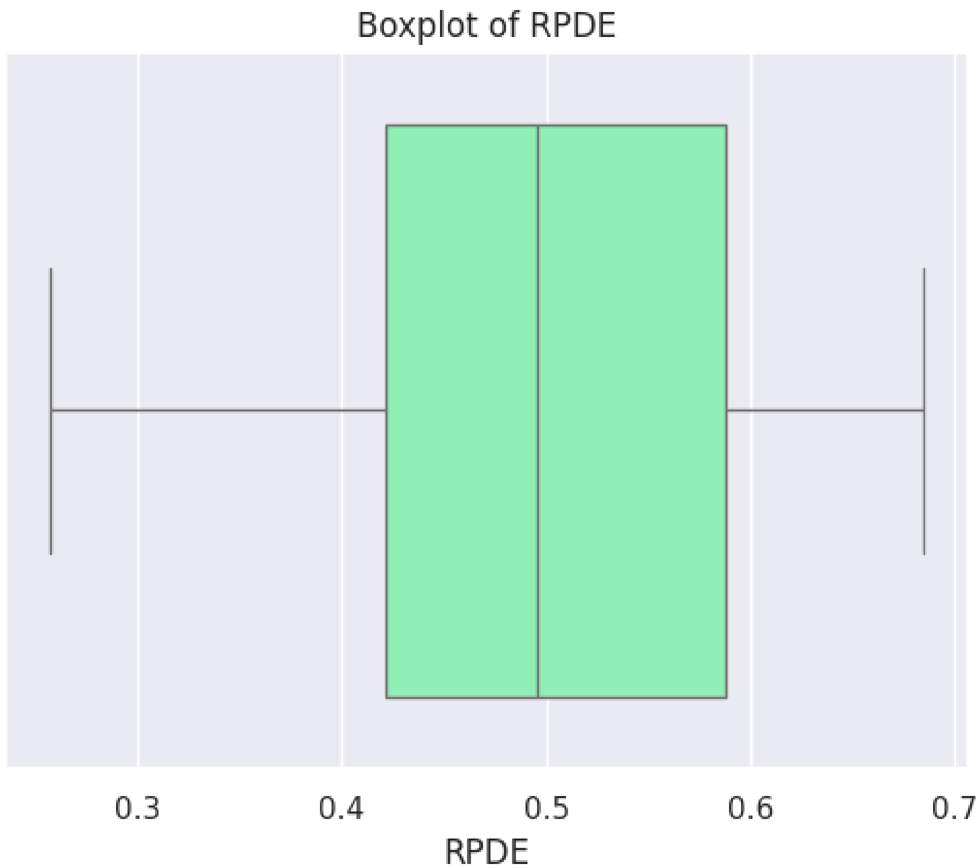


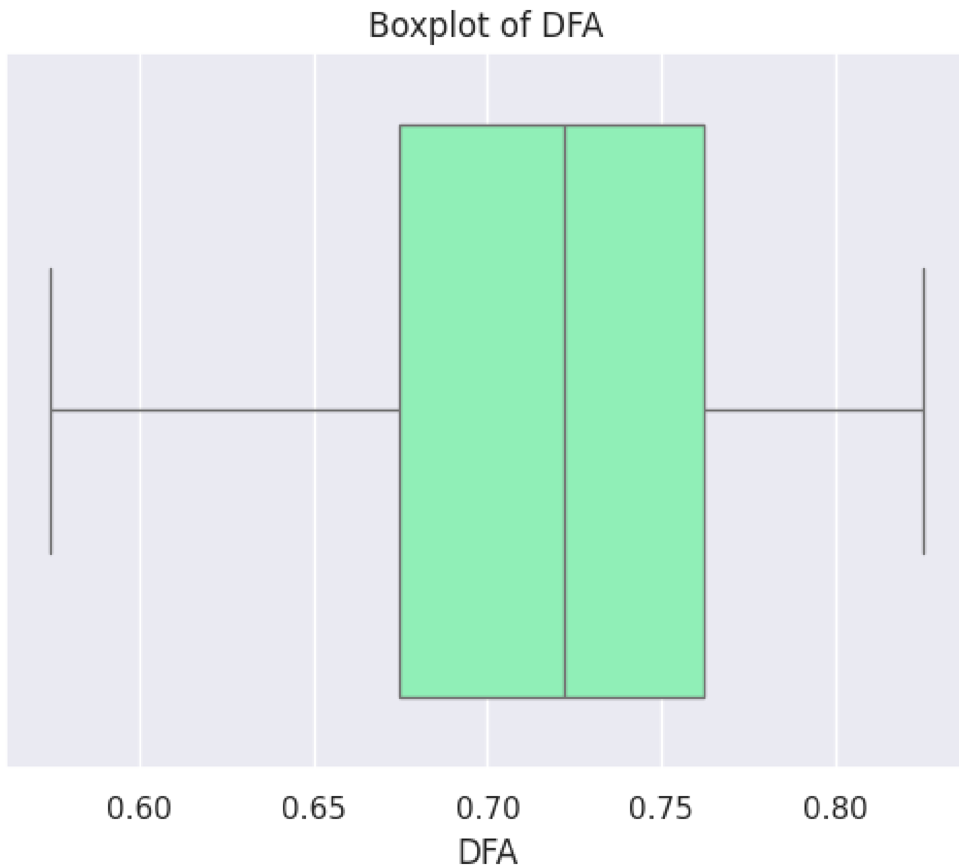
Boxplot of NHR
0.00
0.05
0.10
0.15
0.20
0.25
0.30
NHR

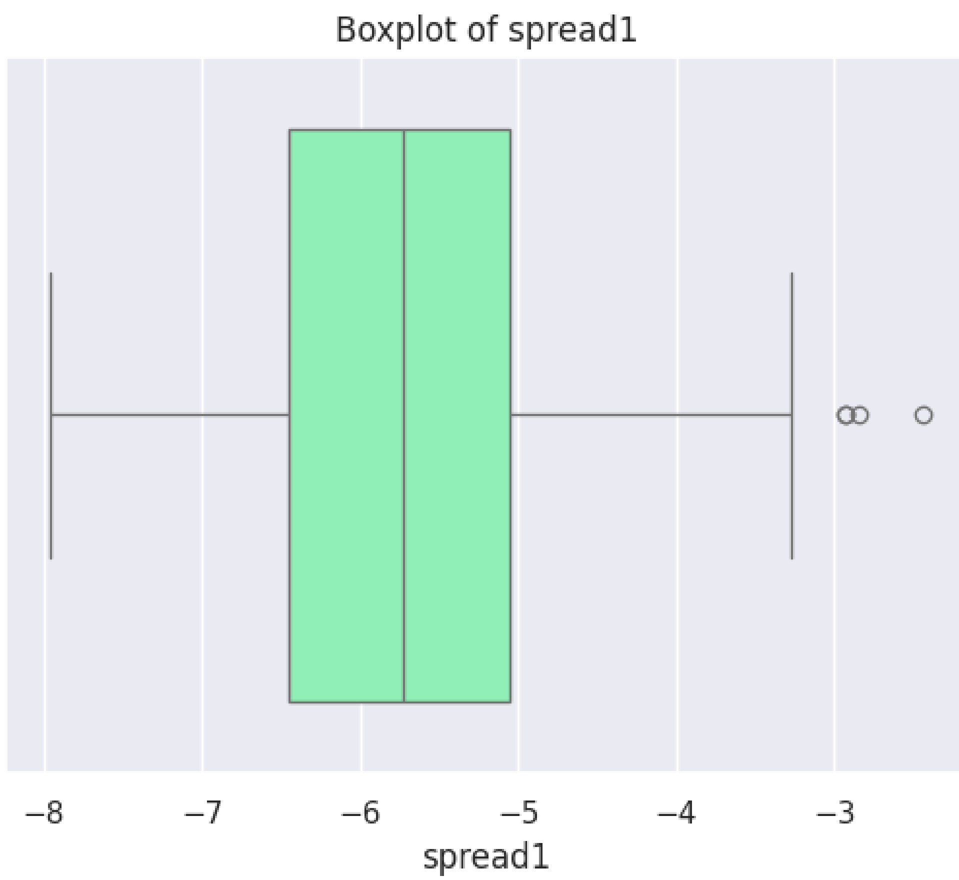

Boxplot of HNR
10
15
20
25
30
HNR

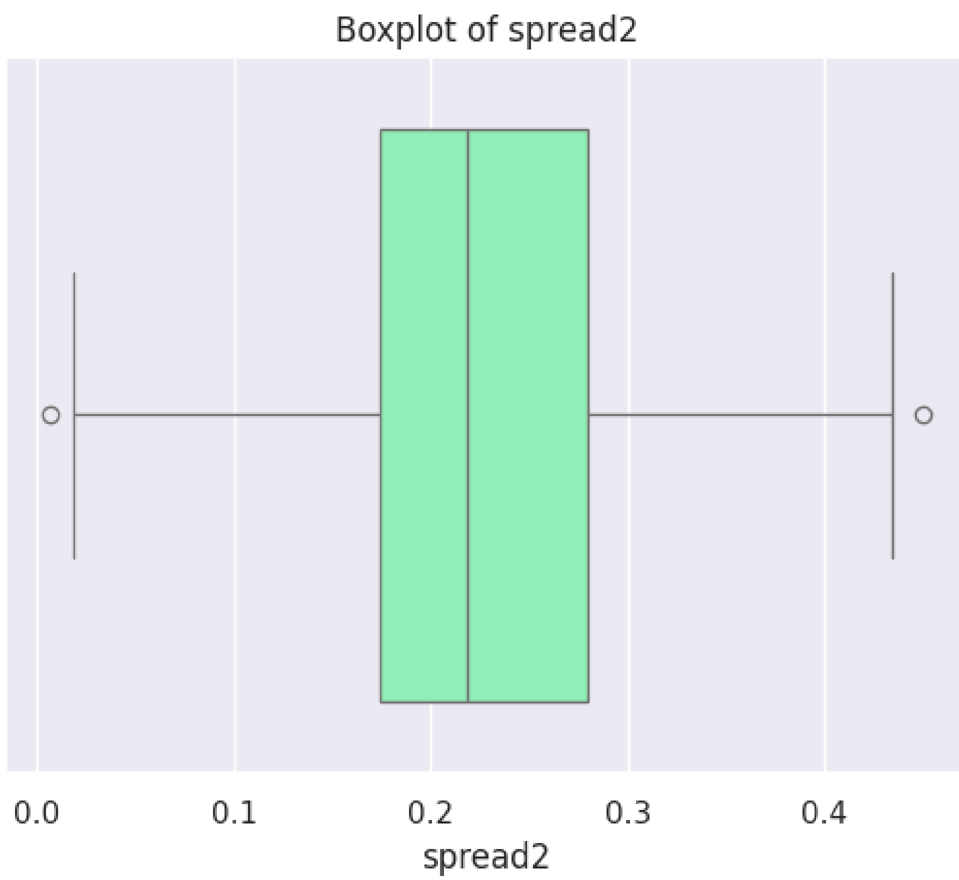


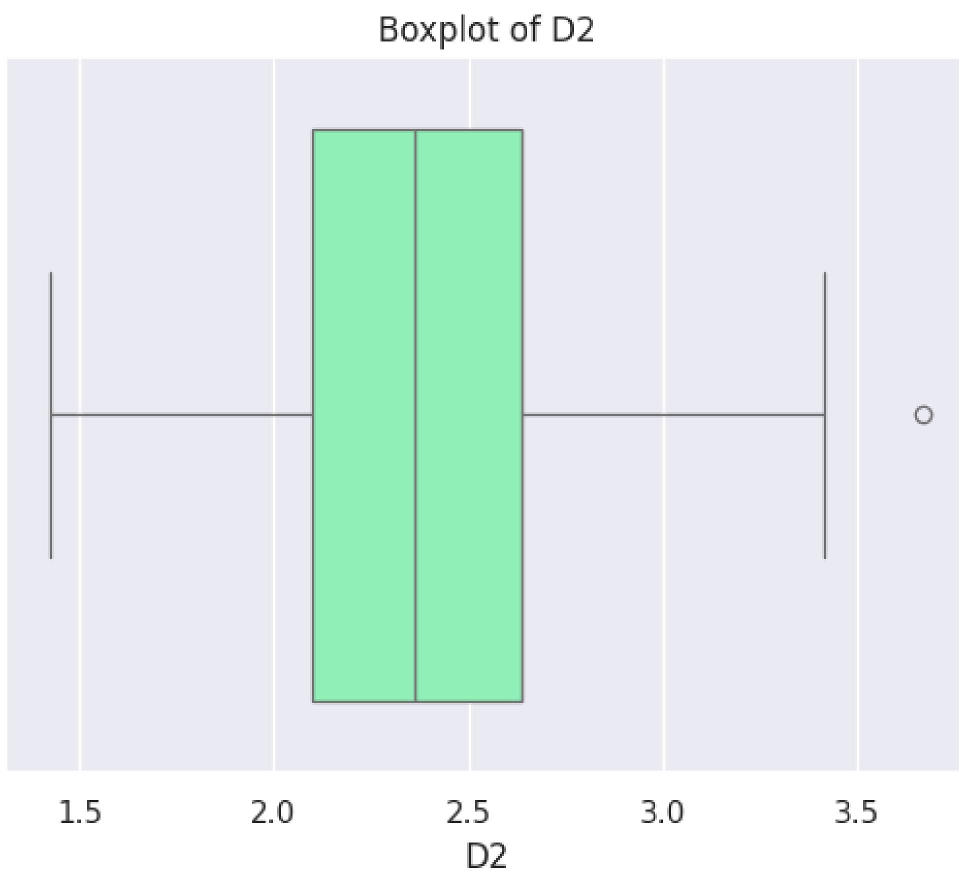
Boxplot of status
0.0
0.2
0.4
0.6
0.8
1.0
status

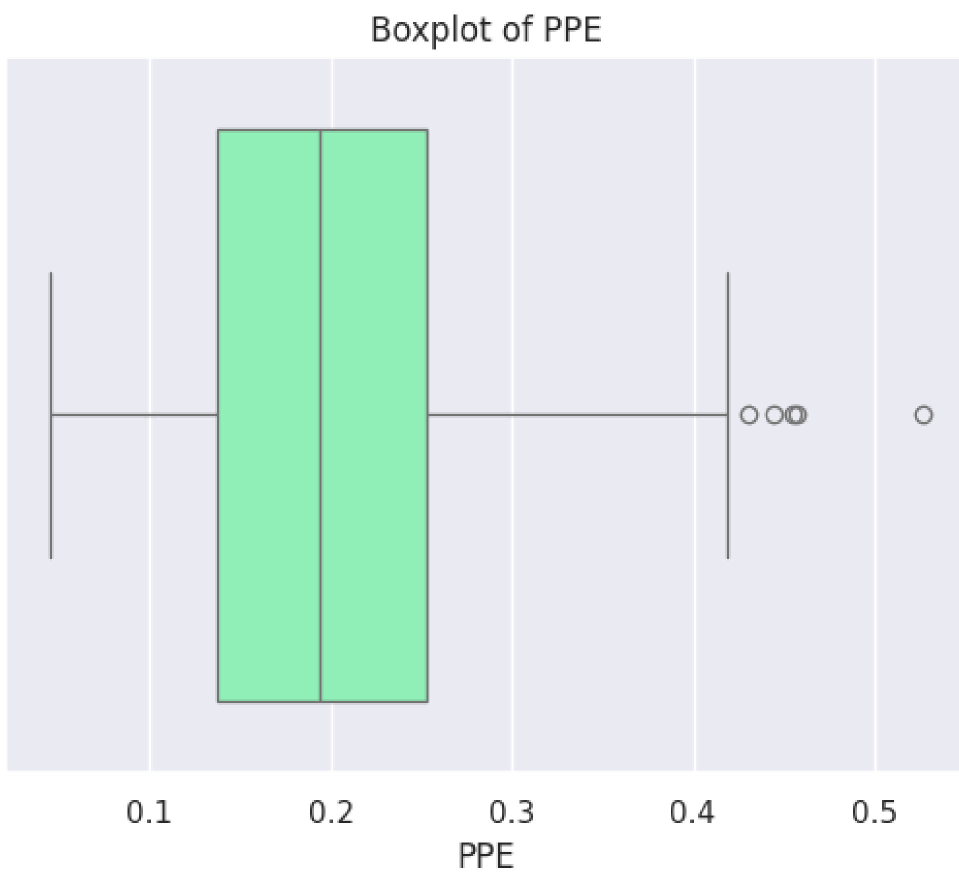

Boxplot of RPDE
0.3
0.4
0.5
0.6
0.7
RPDE


Boxplot of DFA
0.60
0.65
0.70
0.75
0.80
DFA


Boxplot of spread1
−8
−7
−6
−5
−4
−3
spread1


Boxplot of spread2
0.0
0.1
0.2
0.3
0.4
spread2


Boxplot of D2
1.5
2.0
2.5
3.0
3.5
D2



outliers are not treated in this model because,

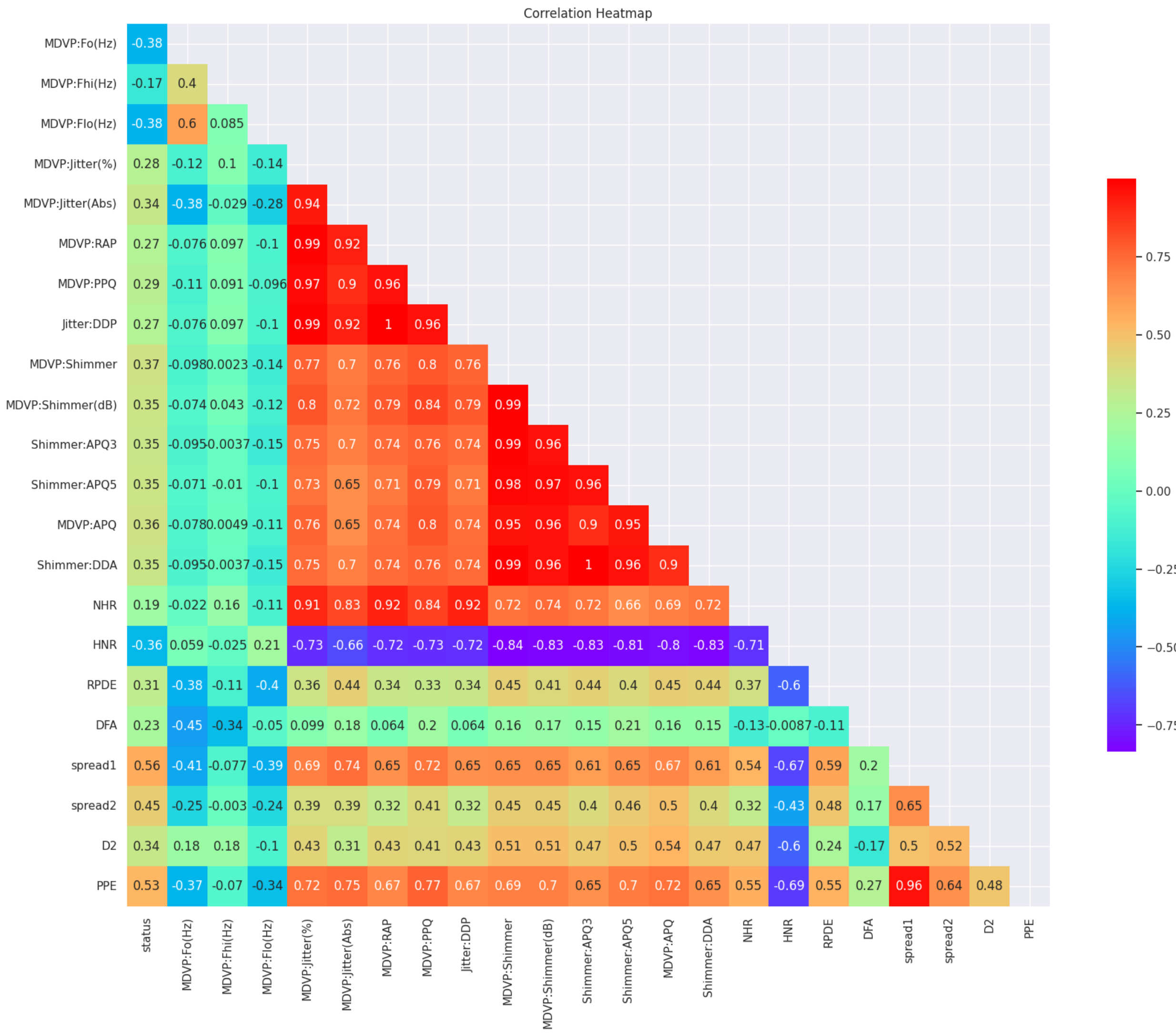
- The dataset is related to Parkinson's disease, where certain features (e.g., voice frequency or amplitude) might naturally exhibit significant variability due to the condition's effects.
- These "outliers" might carry critical information distinguishing patients with Parkinson's from healthy individuals.

Finding correlation and multicolinearity of features and target variable

```
non_numeric_cols =
df.select_dtypes(exclude=['number']).columns.tolist()
print("Non-numeric columns:", non_numeric_cols)

numeric_df = df.select_dtypes(include=['number'])

corr = numeric_df.corr()

if 'status' in corr.columns:
    cols = ['status'] + [col for col in corr.columns if col !=
'status']
    corr = corr.loc[cols, cols]
```

```python
corr = corr.drop(index='status')

mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

plt.figure(figsize=(20, 20))
sns.heatmap(
    corr,
    mask=mask,
    annot=True,
    cmap='rainbow',
    square=True,
    linewidths=0,
    cbar_kws={'shrink': 0.5},
    facecolor='white'
)
plt.title("Correlation Heatmap")
plt.show()
```

```
Non-numeric columns: ['name']
```



According to the above heatmap we have identified there are features that are highly dependent on target variable as well as least depend on the target variable. We also identified some features with multicolinearity.

Spliting dataset into features and target

```
# Split dataset into features and target
x = df.drop(['status', 'name'], axis=1)
y = df['status']
print(f"Shape of feature matrix: {x.shape}")

Shape of feature matrix: (195, 22)

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
# from sklearn.decomposition import PCA
from imblearn.over_sampling import RandomOverSampler
```

```
print(x.shape)
```

```
(195, 22)
```

```
x.head() #Independent Variable
```

{"type":"dataframe","variable_name":"x"}

```
y.head() #Dependent Variable
```

```
0    1
1    1
2    1
3    1
4    1
Name: status, dtype: int64
```

Train Test Split

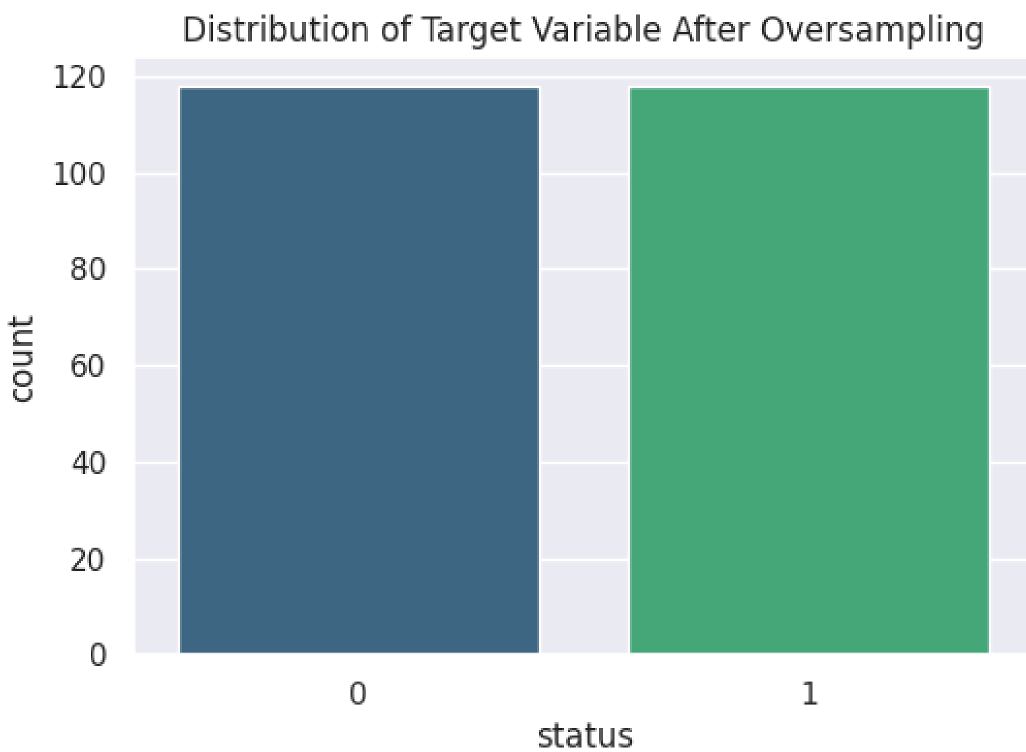
```
# Split data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42, stratify=y)
print(f"Training set size: {x_train.shape}, Test set size:
{x_test.shape}")
```

```
Training set size: (156, 22), Test set size: (39, 22)
```

Using SMOTE to balance the dataset

```
# Balancing Data Using SMOTE
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
x_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
print(f"After Oversampling - Training set size: {x_train_smote.shape},
Class distribution: {y_train_smote.value_counts()}")
plt.figure(figsize=(6, 4))
sns.countplot(x=y_train_smote, palette='viridis')
plt.title("Distribution of Target Variable After Oversampling")
plt.show()
```

```
After Oversampling - Training set size: (236, 22), Class distribution:
status
0    118
1    118
Name: count, dtype: int64
```



Scaling train and test set seperately using s]Standard Scalar

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train_smote)
x_test_scaled = scaler.transform(x_test)
```

we fit and transform only the training data (and not the test data) to prevent data leakage and ensure fair evaluation of the model's performance.By fitting only the training data, we ensure the model has no prior knowledge of the test set.

Applying SelectKBest to find most important features for the model

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif

# Selecting top 8 features
selector = SelectKBest(mutual_info_classif, k=8)
x_train_kbest = selector.fit_transform(x_train_scaled, y_train_smote)
x_test_kbest = selector.transform(x_test_scaled)
x_train_selected = x_train_kbest
x_test_selected = x_test_kbest
print(f"Number of features selected: {x_train_selected.shape[1]}")

Number of features selected: 8
```

The reason to use SelectKBest,

- Selects features with the strongest statistical correlation to the target variable, SelectKBest ensures that only the most important predictors are included, which can improve model performance.
- SelectKBest is easy to implement and interpret
- simplicity, speed, and ability to identify significant features without altering their interpretability

Model Implementation

SVM - SUpport Vector Machine

Hyperprameter Tuning

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, StratifiedKFold,
GridSearchCV

# SVM with Hyperparameter Tuning
param_grid_svm = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}
grid_svm = GridSearchCV(SVC(), param_grid_svm, scoring='accuracy',
cv=StratifiedKFold(n_splits=5))
grid_svm.fit(x_train_selected, y_train_smote)
best_svm = grid_svm.best_estimator_
print(f"Best Parameters for SVM: {grid_svm.best_params_}")

Best Parameters for SVM: {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
```

Support Vector Machine (SVM) is a supervised learning algorithm widely used for classification problems. It is particularly effective in scenarios where the data is not linearly separable. Reason to select SVM are,

- SVM works well in high-dimensional spaces, making it ideal for datasets with numerous features.

- SVM performs efficiently with limited data points, which is beneficial when data availability is constrained

- SVM was chosen because it can effectively classify the Parkinson's disease dataset, where features may exhibit non-linear relationships.

KNN- K nearest Neighbors Classification

Hyperparameter Tuning

```
from sklearn.neighbors import KNeighborsClassifier

# KNN with Hyperparameter Tuning
param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'p': [1, 2]
}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn,
scoring='accuracy', cv=StratifiedKFold(n_splits=5))
grid_knn.fit(x_train_selected, y_train_smote)
best_knn = grid_knn.best_estimator_

print(f"Best Parameters for KNN: {grid_knn.best_params_}")
```

```
Best Parameters for KNN: {'n_neighbors': 7, 'p': 1, 'weights':
'distance'}
```

K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm that classifies data points based on the majority class of their nearest neighbors in feature space.Reason to select KNN,

- KNN is easy to implement and understand, making it an excellent baseline classifier for comparison.
- KNN is computationally efficient for smaller datasets like ours and can provide reliable predictions.
- KNN was used to evaluate its performance as a straightforward and interpretable alternative to more complex classifiers like SVM

Reason to select GridSearchCV,

GridSearch is an approach to hyperparameter optimization, where all possible combinations of a specified parameter grid are evaluated to find the best-performing parameter set.

- GridSearch ensures that all combinations of hyperparameters are evaluated, providing the best configuration for model performance.
- By optimizing parameters like C, kernel, and gamma (for SVM) or k, weights, and distance metric (for KNN), GridSearch helps achieve the highest possible accuracy for the given dataset.
- By tuning parameters for both models, GridSearch ensures a fair and unbiased comparison between SVM and KNN.

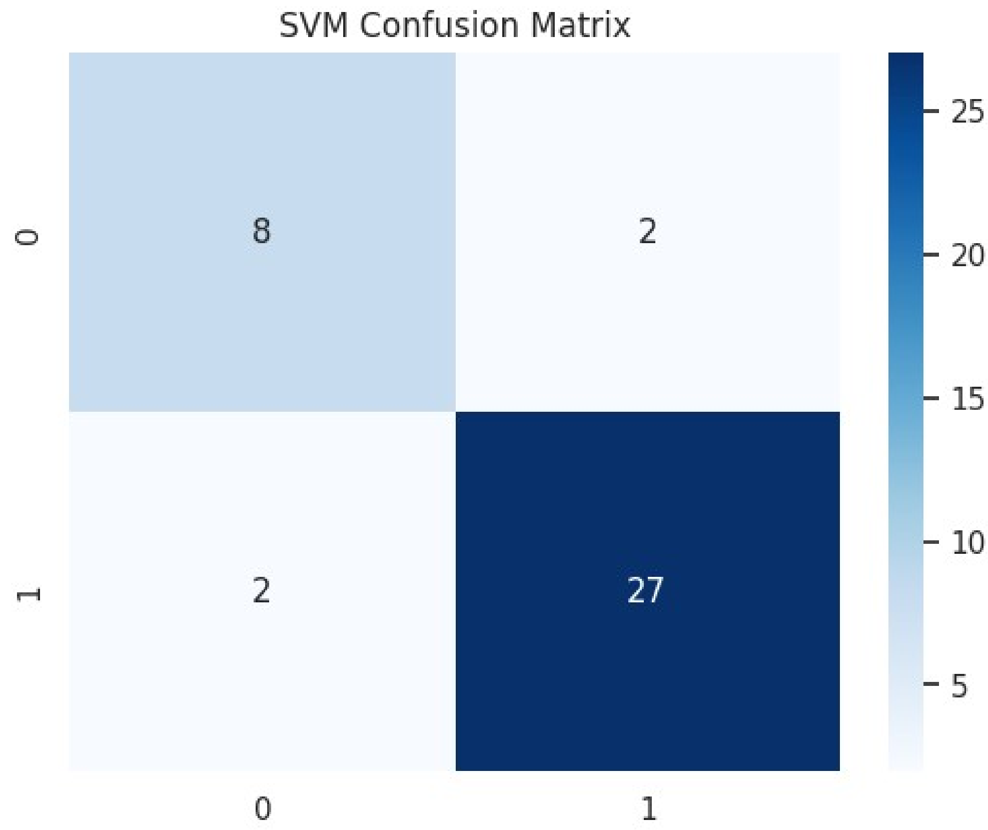
Model Evaluation and Discussion

```
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
```

SVM evaluation

```
# SVM Evaluation
y_pred_svm = best_svm.predict(x_test_selected)
print("\nSVM Classification Report:")
print(classification_report(y_test, y_pred_svm))
sns.heatmap(confusion_matrix(y_test, y_pred_svm), annot=True,
cmap='Blues', fmt='d')
plt.title("SVM Confusion Matrix")
plt.show()
```

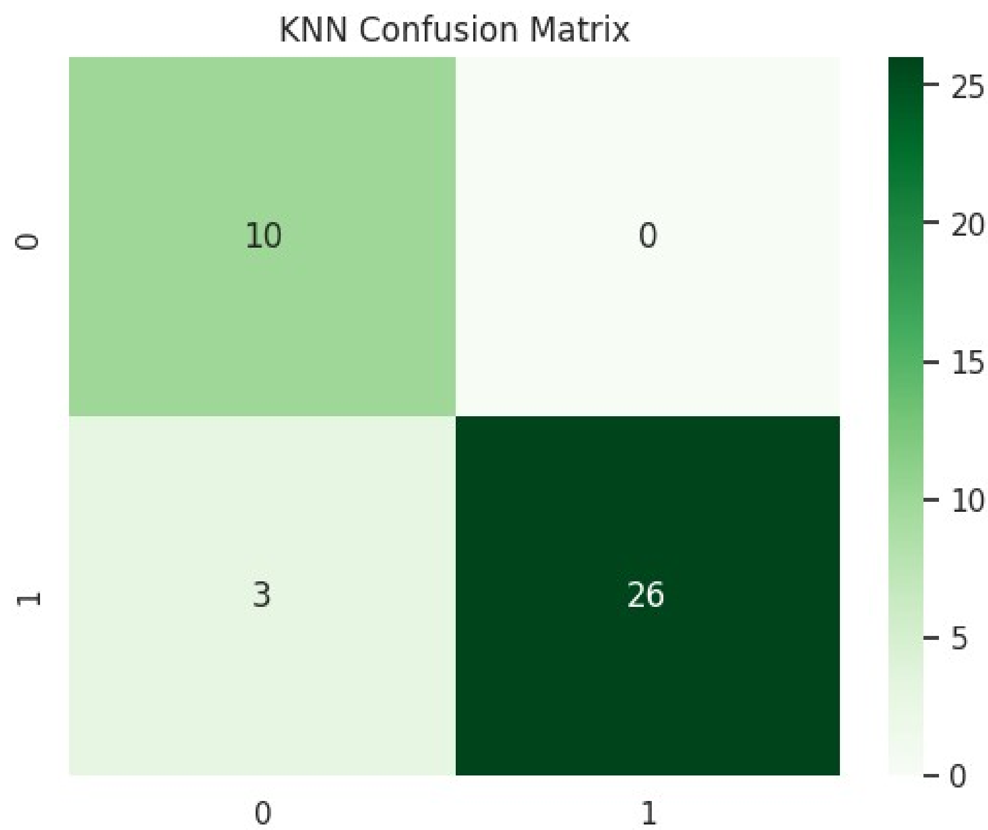
```
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.80      0.80        10
           1       0.93      0.93      0.93        29

    accuracy                           0.90        39
   macro avg       0.87      0.87      0.87        39
weighted avg       0.90      0.90      0.90        39
```



KNN Evaluation

```
# KNN Evaluation
y_pred_knn = best_knn.predict(x_test_selected)
print("\nKNN Classification Report:")
print(classification_report(y_test, y_pred_knn))
sns.heatmap(confusion_matrix(y_test, y_pred_knn), annot=True,
cmap='Greens', fmt='d')
plt.title("KNN Confusion Matrix")
plt.show()
```

```
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.77      1.00      0.87        10
           1       1.00      0.90      0.95        29

    accuracy                           0.92        39
   macro avg       0.88      0.95      0.91        39
weighted avg       0.94      0.92      0.93        39
```



Model Evaluation and finding best model

```
# Compare Models
train_accuracy_svm = grid_svm.best_score_
test_accuracy_svm = accuracy_score(y_test, y_pred_svm)

train_accuracy_knn = grid_knn.best_score_
test_accuracy_knn = accuracy_score(y_test, y_pred_knn)

print("\nModel Comparison:")
print(f"SVM: Training Accuracy: {train_accuracy_svm:.2f}, Testing
Accuracy: {test_accuracy_svm:.2f}")
print(f"KNN: Training Accuracy: {train_accuracy_knn:.2f}, Testing
Accuracy: {test_accuracy_knn:.2f}")
```

```
Model Comparison:
SVM: Training Accuracy: 0.94, Testing Accuracy: 0.90
KNN: Training Accuracy: 0.95, Testing Accuracy: 0.92
```

We have used multiple evaluation metrics to analyze and compare the performance of the SVM and KNN classifiers.Evaluation Metrics Used are,

1. Accuracy

- Accuracy measures the proportion of correctly predicted instances out of the total instances.
- For SVM, the training accuracy was 0.94, while the testing accuracy was 0.90. For KNN, the training accuracy was 0.95, and the testing accuracy was 0.92.

2. Classification Report

- A detailed classification report was generated for both models, including precision, recall, and F1-score.This provided insights into how well the models handled each class.

3. Confusion Matrix

- The confusion matrix was visualized to evaluate the models' performance in distinguishing between the two classes.
- It helped identify whether the models had a bias toward any specific class and highlighted any misclassifications.

Comparing Models

- The SVM model achieved a high training accuracy of 0.94 but slightly lower testing accuracy of 0.90. This indicates that the model performed well but might slightly overfit the training data.
- The KNN model had a training accuracy of 0.95 and a higher testing accuracy than SVM of 0.92. This demonstrates that KNN was better at generalizing to the unseen testing data, likely due to its non-parametric nature and simplicity.

Discussion

The performance metrics indicate that both SVM and KNN are highly effective for the given dataset. However, there are some important observations

---

- The SVM model achieved slightly better training accuracy, showing its ability to capture complex patterns in the data.
- However, the lower testing accuracy suggests mild overfitting, likely due to the optimization of hyperparameters like C and gamma.
- KNN had a slightly higher training accuracy and outperformed SVM on the testing set with an accuracy of 0.92.Its simplicity and non-parametric nature made it less prone to overfitting.
- SVM's ability to capture complex relationships is advantageous, but its sensitivity to hyperparameters requires careful tuning.

- KNN provided a more generalizable solution with a simpler implementation, making it an excellent choice for this dataset.

Conclusion

```
from sklearn.model_selection import cross_val_score

scores_svm = cross_val_score(best_svm, x_train_kbest, y_train_smote, cv=5, scoring='accuracy')
scores_knn = cross_val_score(best_knn, x_train_kbest, y_train_smote, cv=5, scoring='accuracy')

print("SVM Cross-Validation Accuracy:", scores_svm.mean())
print("KNN Cross-Validation Accuracy:", scores_knn.mean())

SVM Cross-Validation Accuracy: 0.9407801418439716
KNN Cross-Validation Accuracy: 0.9450354609929077
```

- Cross-validation was conducted to evaluate the consistency of model performance across different data splits. Both SVM and KNN achieved high cross-validation accuracies of 94.08% and 94.50%, respectively, indicating strong generalization capabilities.
- While both models showed comparable performance, KNN demonstrated a marginally higher cross-validation accuracy, suggesting slightly better generalization to unseen data. However, the difference is negligible, and both models are well-suited.

```
# Conclution
if test_accuracy_svm > test_accuracy_knn:
    print("SVM is the better model.")
else:
    print("KNN is the better model.")

KNN is the better model.
```

Based on the results,

- KNN is the better-performing model in this scenario, as it achieved higher testing accuracy, indicating better generalization to unseen data.
- SVM remains a strong candidate for datasets with more complex relationships or in situations where interpretability of decision boundaries is critical.

References

1. Marton H. (2019). You Could Have Parkinson's Disease Symptoms in Your 30s or 40s and not Know It. Available online at: https://www.healthdirect.gov.au/blog/parkinsons-disease-symptoms-in-your-30s-40s

2. S. Galli, "Train In Data," Imbalanced Data, Machine Learning, 20 March 2023. [Online]. Available: https://www.blog.trainindata.com/oversampling-techniques-for-imbalanced-data/.

3. C. G. Goetz, "Cold Spring Harb Perspect Med," The History of Parkinson's Disease: Early Clinical Descriptions and Neurological Therapies, 1 sep 2011. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC3234454/#:~:text=Parkinson's%20disease%20was%20first%20medically,earlier%20descriptions%20(Parkinson%201817)..

4. J. B. J. S.-l. S. J. P. R Lafuente, "Clinical Biomechanics," Design and test of neural networks and statistical classifiers in computer-aided movement analysis: a case study on gait analysis, 25 August 1998. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S026800339700082X.

5. C. D. ,. J. F. Jie Mei, Machine Learning for the Diagnosis of Parkinson's Disease: A Review of Literature, 6 May 2021. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8134676/.

6. S. M. R. S. h. S. Radha N, Parkinson’s Disease Detection using Machine Learning Techniques, 2021. [Online]. Available: https://www.revistaclinicapsicologica.com/archivesarticle.php?id=494.

7. Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection', Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM. BioMedical Engineering OnLine 2007, 6:23 (26 June 2007)

```
!jupyter nbconvert --to html GP_13.ipynb
```

```
[NbConvertApp] Converting notebook GP_13.ipynb to html
[NbConvertApp] WARNING | Alternative text is missing on 51 image(s).
[NbConvertApp] Writing 1998674 bytes to GP_13.html
```