# PARKINSON'S DISEASE

WHAT IS PARKINSON'S DISEASE? Parkinson's disease is a progressive disorder that affects the nervous system and the parts of the body controlled by the nerves. Symptoms start slowly. The first symptom may be a barely noticeable tremor in just one hand. Tremors are common, but the disorder also may cause stiffness or slowing of movement.

In the early stages of Parkinson's disease, your face may show little or no expression. Your arms may not swing when you walk. Your speech may become soft or slurred. Parkinson's disease symptoms worsen as your condition progresses over time. **DESCRIPTION:** 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

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
```

```
1  disease=pd.read_csv("/content/parkinsons data.csv")
```

```
1  disease.head()
```

| | name | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RAP | MDVP:PPQ | Jitter:DDP | MDVP:Shimm |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | phon_R01_S01_1 | 119.992 | 157.302 | 74.997 | 0.00784 | 0.00007 | 0.00370 | 0.00554 | 0.01109 | 0.043 |
| **1** | phon_R01_S01_2 | 122.400 | 148.650 | 113.819 | 0.00968 | 0.00008 | 0.00465 | 0.00696 | 0.01394 | 0.061 |
| **2** | phon_R01_S01_3 | 116.682 | 131.111 | 111.555 | 0.01050 | 0.00009 | 0.00544 | 0.00781 | 0.01633 | 0.052 |
| **3** | phon_R01_S01_4 | 116.676 | 137.871 | 111.366 | 0.00997 | 0.00009 | 0.00502 | 0.00698 | 0.01505 | 0.054 |
| **4** | phon_R01_S01_5 | 116.014 | 141.781 | 110.655 | 0.01284 | 0.00011 | 0.00655 | 0.00908 | 0.01966 | 0.064 |

5 rows × 24 columns

```
1 disease.shape
```

```
(195, 24)
```

```
1 disease.tail()
```

| | name | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RAP | MDVP:PPQ | Jitter:DDP | MDVP:Shi |
|---|---|---|---|---|---|---|---|---|---|---|
| **190** | phon_R01_S50_2 | 174.188 | 230.978 | 94.261 | 0.00459 | 0.00003 | 0.00263 | 0.00259 | 0.00790 | 0.0 |
| **191** | phon_R01_S50_3 | 209.516 | 253.017 | 89.488 | 0.00564 | 0.00003 | 0.00331 | 0.00292 | 0.00994 | 0.0 |
| **192** | phon_R01_S50_4 | 174.688 | 240.005 | 74.287 | 0.01360 | 0.00008 | 0.00624 | 0.00564 | 0.01873 | 0.0 |
| **193** | phon_R01_S50_5 | 198.764 | 396.961 | 74.904 | 0.00740 | 0.00004 | 0.00370 | 0.00390 | 0.01109 | 0.0 |
| **194** | phon_R01_S50_6 | 214.289 | 260.277 | 77.973 | 0.00567 | 0.00003 | 0.00295 | 0.00317 | 0.00885 | 0.0 |

5 rows × 24 columns

```
1 Start coding or generate with AI.
```

```
1 disease.columns
```

```
Index(['name', 'MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

```
1 disease.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
```

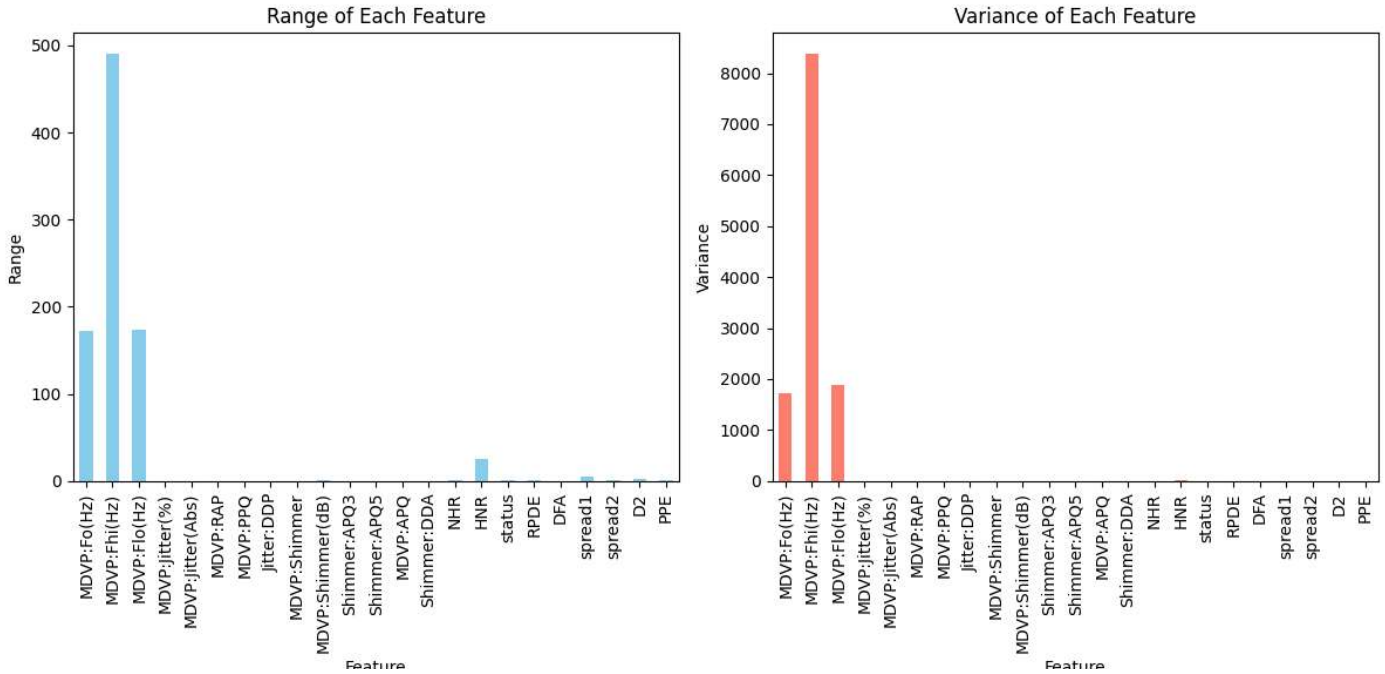
```
1 disease.describe()
```

| | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RAP | MDVP:PPQ | Jitter:DDP | MDVP:Shimmer | MDVP:S |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | |
| **mean** | 154.228641 | 197.104918 | 116.324631 | 0.006220 | 0.000044 | 0.003306 | 0.003446 | 0.009920 | 0.029709 | |
| **std** | 41.390065 | 91.491548 | 43.521413 | 0.004848 | 0.000035 | 0.002968 | 0.002759 | 0.008903 | 0.018857 | |
| **min** | 88.333000 | 102.145000 | 65.476000 | 0.001680 | 0.000007 | 0.000680 | 0.000920 | 0.002040 | 0.009540 | |
| **25%** | 117.572000 | 134.862500 | 84.291000 | 0.003460 | 0.000020 | 0.001660 | 0.001860 | 0.004985 | 0.016505 | |
| **50%** | 148.790000 | 175.829000 | 104.315000 | 0.004940 | 0.000030 | 0.002500 | 0.002690 | 0.007490 | 0.022970 | |
| **75%** | 182.769000 | 224.205500 | 140.018500 | 0.007365 | 0.000060 | 0.003835 | 0.003955 | 0.011505 | 0.037885 | |
| **max** | 260.105000 | 592.030000 | 239.170000 | 0.033160 | 0.000260 | 0.021440 | 0.019580 | 0.064330 | 0.119080 | |

8 rows × 23 columns

```
1 disease=disease.drop(columns=['name'])
```

```
df = pd.DataFrame(disease)

# Calculating the range and variance for each feature
ranges = df.max() - df.min()
variances = df.var()

# Plotting the range for each feature
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
ranges.plot(kind='bar', color='skyblue')
plt.title('Range of Each Feature')
plt.ylabel('Range')
plt.xlabel('Feature')

# Plotting the variance for each feature
plt.subplot(1, 2, 2)
variances.plot(kind='bar', color='salmon')
plt.title('Variance of Each Feature')
plt.ylabel('Variance')
plt.xlabel('Feature')

plt.tight_layout()
plt.show()
```




```
disease.shape
```

```
(195, 23)
```

```
null_values=disease.isna().sum()
print(null_values)
```

```
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
```

```
print(disease.duplicated().sum())
```

```
0
```

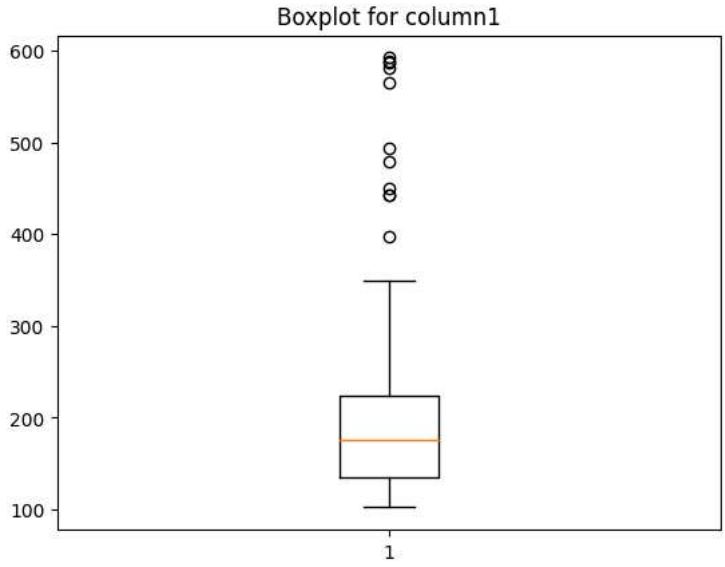
```
data_type=disease.dtypes
print(data_type)
```

```
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
```

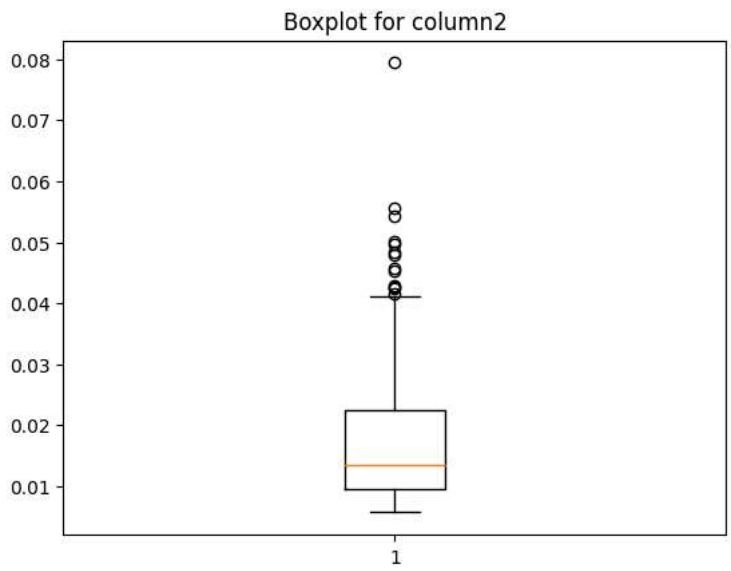
```
missing_values = disease.isnull().sum()
print(missing_values)
```

```
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
```

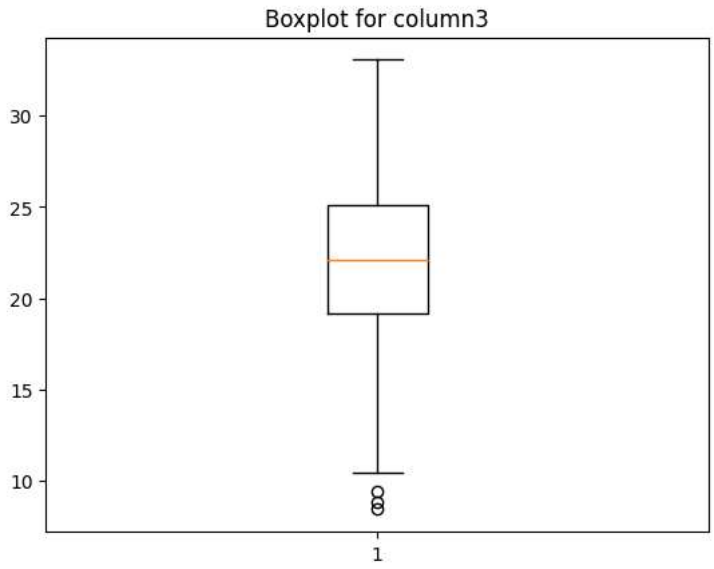
```
plt.boxplot(disease['MDVP:Fhi(Hz)'])
plt.title('Boxplot for column1')
plt.show()
```



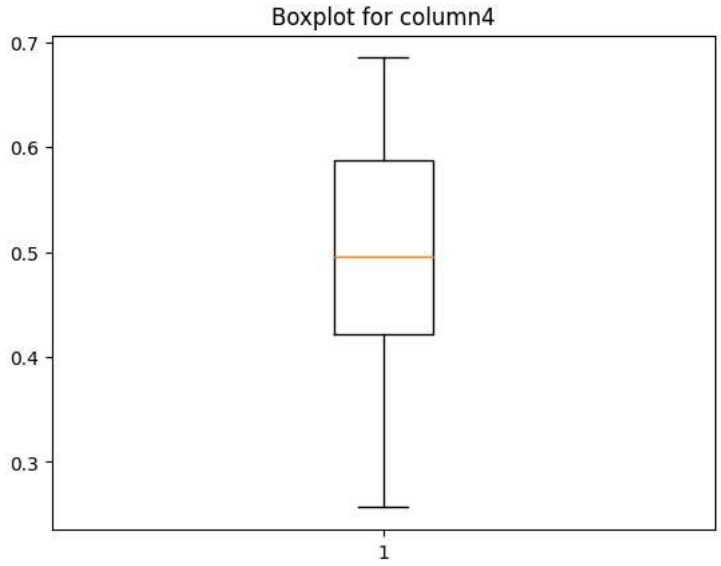

```
1
2 plt.boxplot(disease['Shimmer:APQ5'])
3 plt.title('Boxplot for column2')
4 plt.show()
5
6
7
```



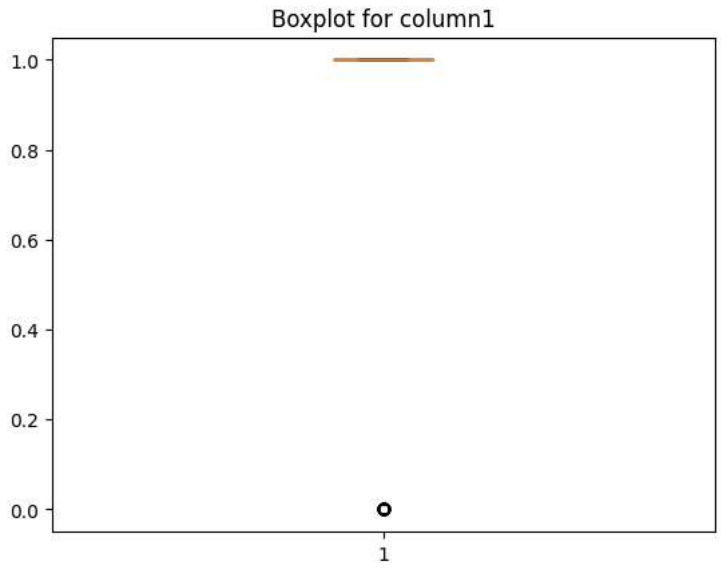

```
1 plt.boxplot(disease['HNR'])
2 plt.title('Boxplot for column3')
3 plt.show()
```



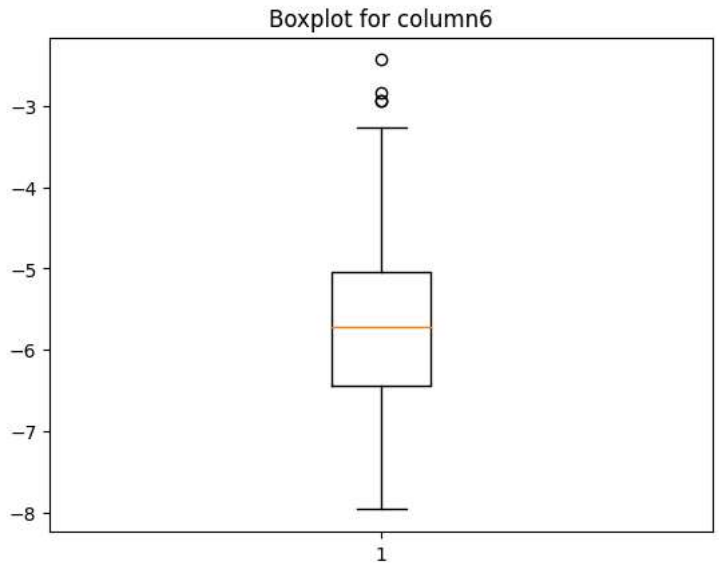

```
plt.boxplot(disease['RPDE'])
plt.title('Boxplot for column4')
plt.show()
```



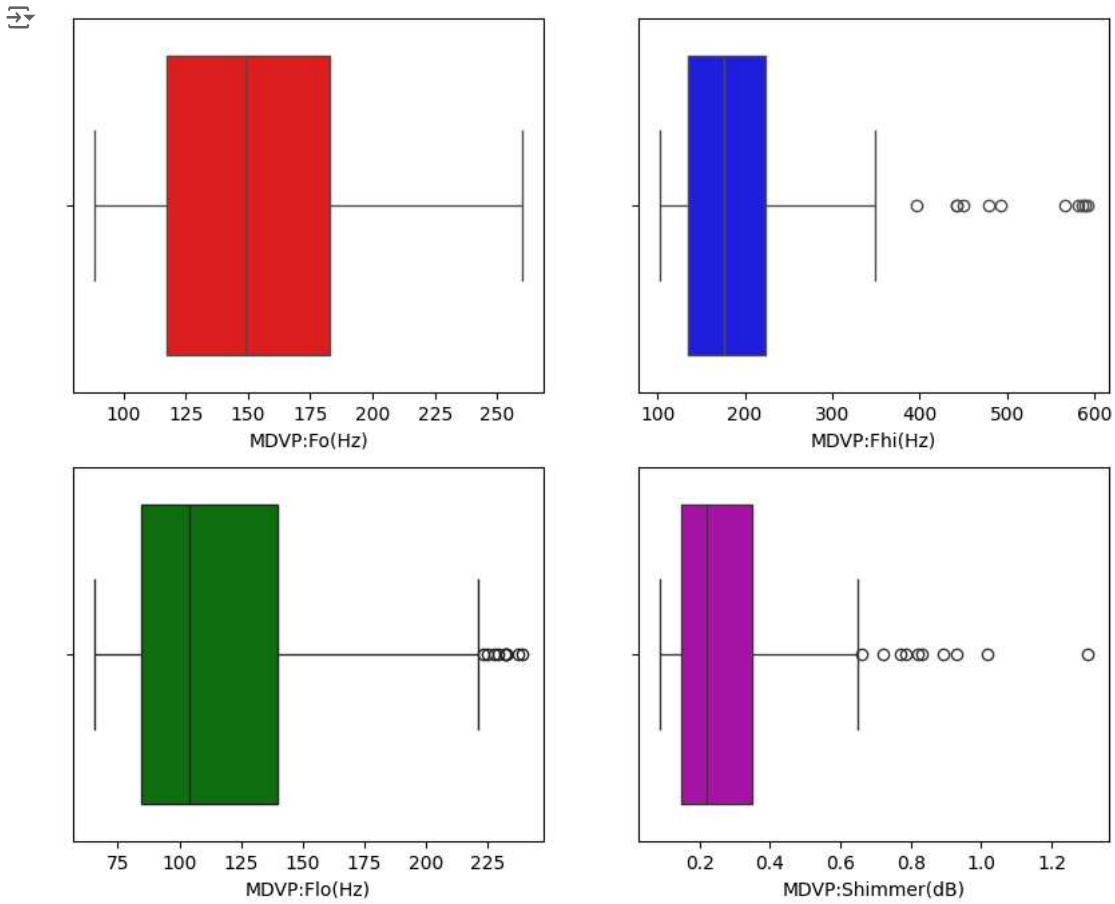

```
plt.boxplot(disease['status'])
plt.title('Boxplot for column1')
plt.show()
```



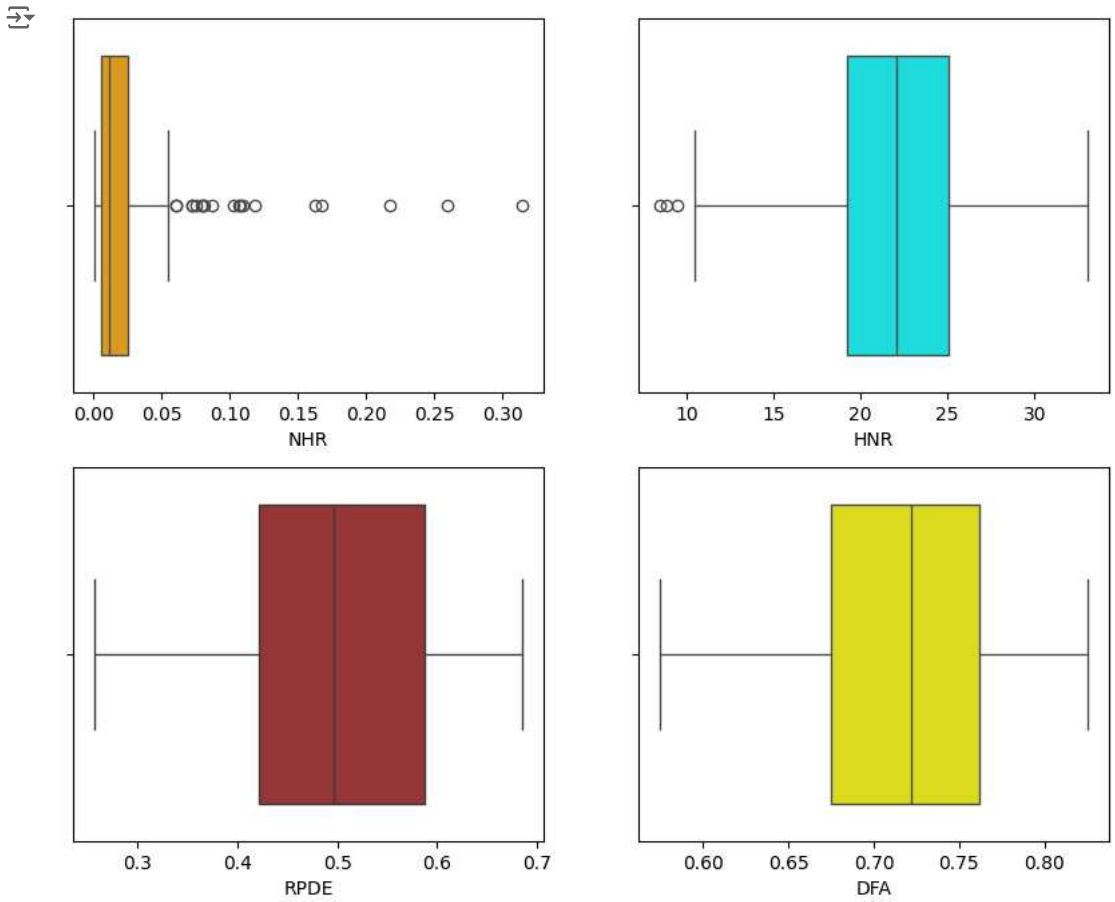

```
plt.boxplot(disease['spread1'])
plt.title('Boxplot for column6')
plt.show()
```



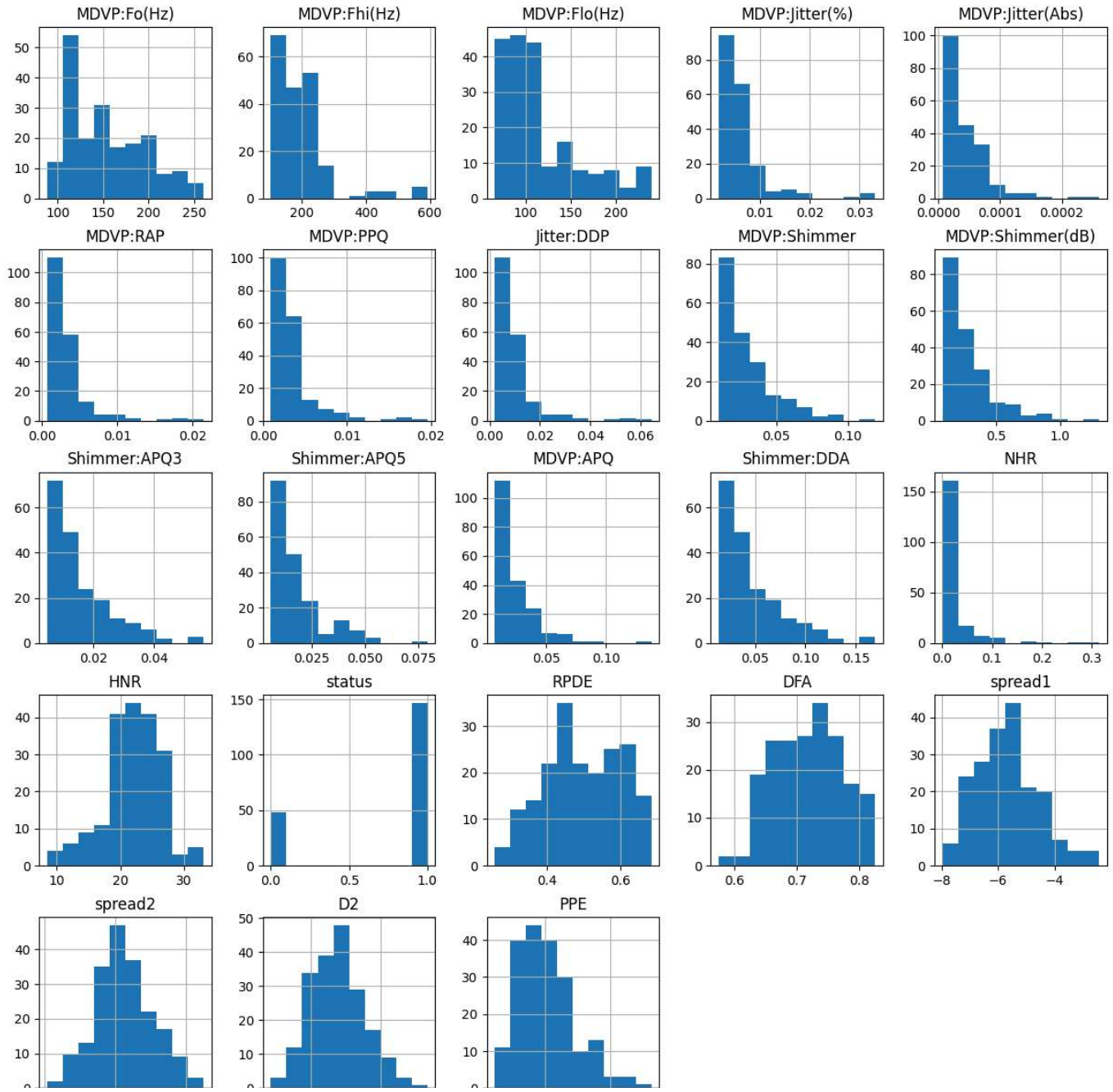

```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(10, 8))
sns.boxplot(disease, x = 'MDVP:Fo(Hz)', ax = ax1, color = 'r')
sns.boxplot(disease, x = 'MDVP:Fhi(Hz)', ax = ax2, color = 'b')
sns.boxplot(disease, x = 'MDVP:Flo(Hz)', ax = ax3, color = 'g')
sns.boxplot(disease, x = 'MDVP:Shimmer(dB)', ax = ax4, color = 'm')
plt.show()
```



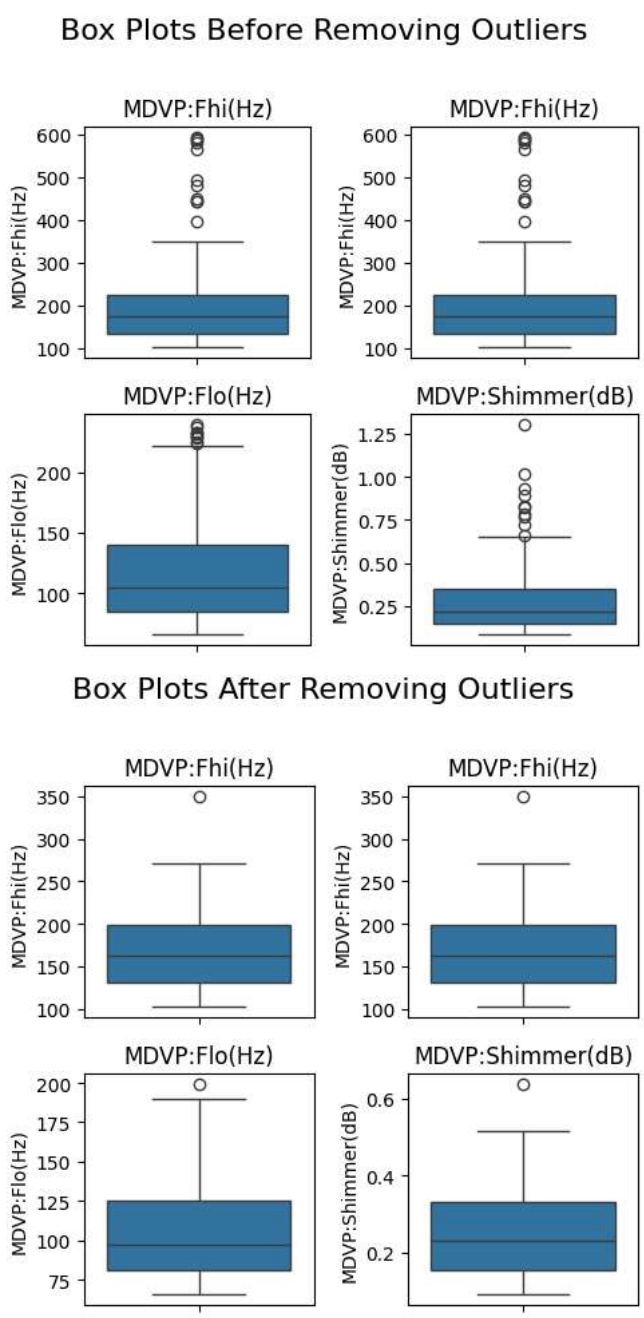

```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(10, 8))
sns.boxplot(disease, x = 'NHR', ax = ax1, color = 'orange')
sns.boxplot(disease, x = 'HNR', ax = ax2, color = 'cyan')
sns.boxplot(disease, x = 'RPDE', ax = ax3, color = 'brown')
sns.boxplot(disease, x = 'DFA', ax = ax4, color = 'yellow')
plt.show()
```




```
1 disease.hist(figsize=(15,15));
```


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

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats


df = pd.DataFrame(disease)

# Visualize data before removing outliers using box plots
plt.figure(figsize=(5,5))
plt.suptitle('Box Plots Before Removing Outliers', fontsize=16)

for i, col in enumerate(['MDVP:Fhi(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Shimmer(dB)']):


    plt.subplot(2, 2, i+1)
    sns.boxplot(y=df[col])
    plt.title(f'{col}')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# Detect and remove outliers using Z-score
def remove_outliers(disease):

    Q1 = disease.quantile(0.25)
    Q3 = disease.quantile(0.75)
    IQR = Q3 - Q1


    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR


    df_no_outliers = disease[~((disease < lower_bound) | (disease > upper_bound)).any(axis=1)]
    return df_no_outliers

# Call the function to remove outliers and create df_no_outliers
df_no_outliers = remove_outliers(disease) # Call the function here

# Visualize data after removing outliers using box plots
plt.figure(figsize=(5,5))
plt.suptitle('Box Plots After Removing Outliers', fontsize=16)

for i, col in enumerate(['MDVP:Fhi(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Shimmer(dB)']):
    plt.subplot(2, 2, i+1)
    sns.boxplot(y=df_no_outliers[col])
    plt.title(f'{col}')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```




Z-SCORE z = ( x - μ ) / σ WHERE, z = Z-score x = the value being evaluated μ = the mean σ = the standard deviation

x = 57 μ = 52 σ = 4 You would use the variables in the formula:

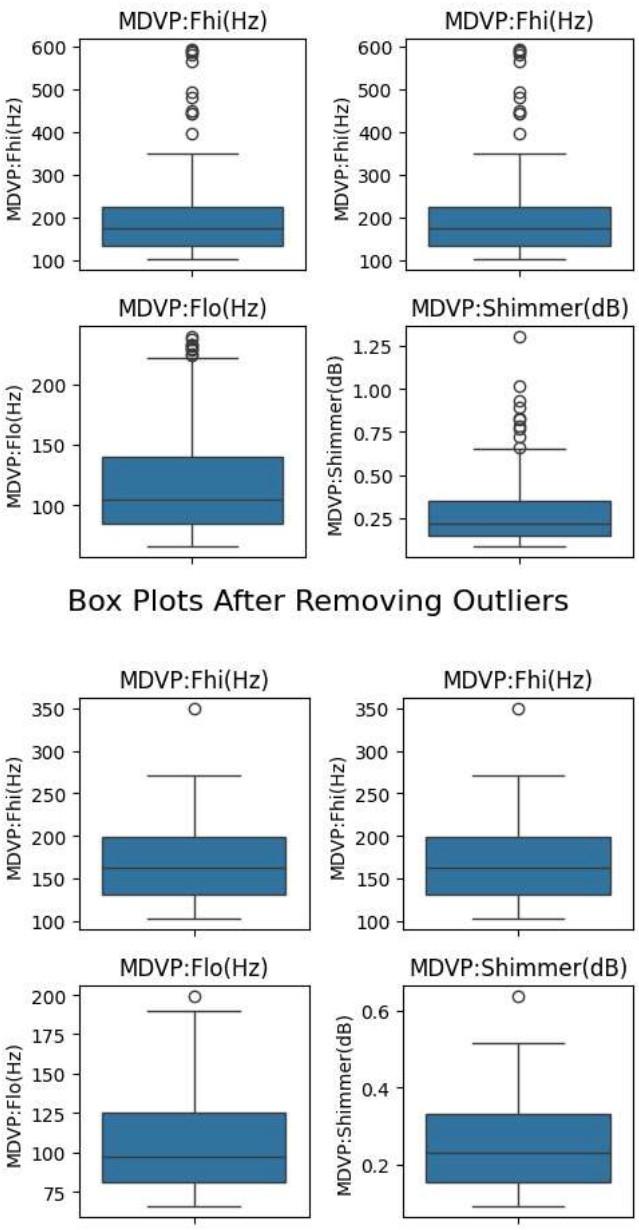
z = ( 57 - 52 ) / 4 z = 1.25 So, your selected value has a z-score that indicates it is 1.25 standard deviations from the mean.

```
from scipy import stats

z_scores = stats.zscore(disease.select_dtypes(include=[float, int]))
abs_z_scores = np.abs(z_scores)
outliers = (abs_z_scores > 3).all(axis=1)
print(disease[outliers])
disease = disease[~outliers]
print("Data without outliers (Z-score):")
print(outliers)
```

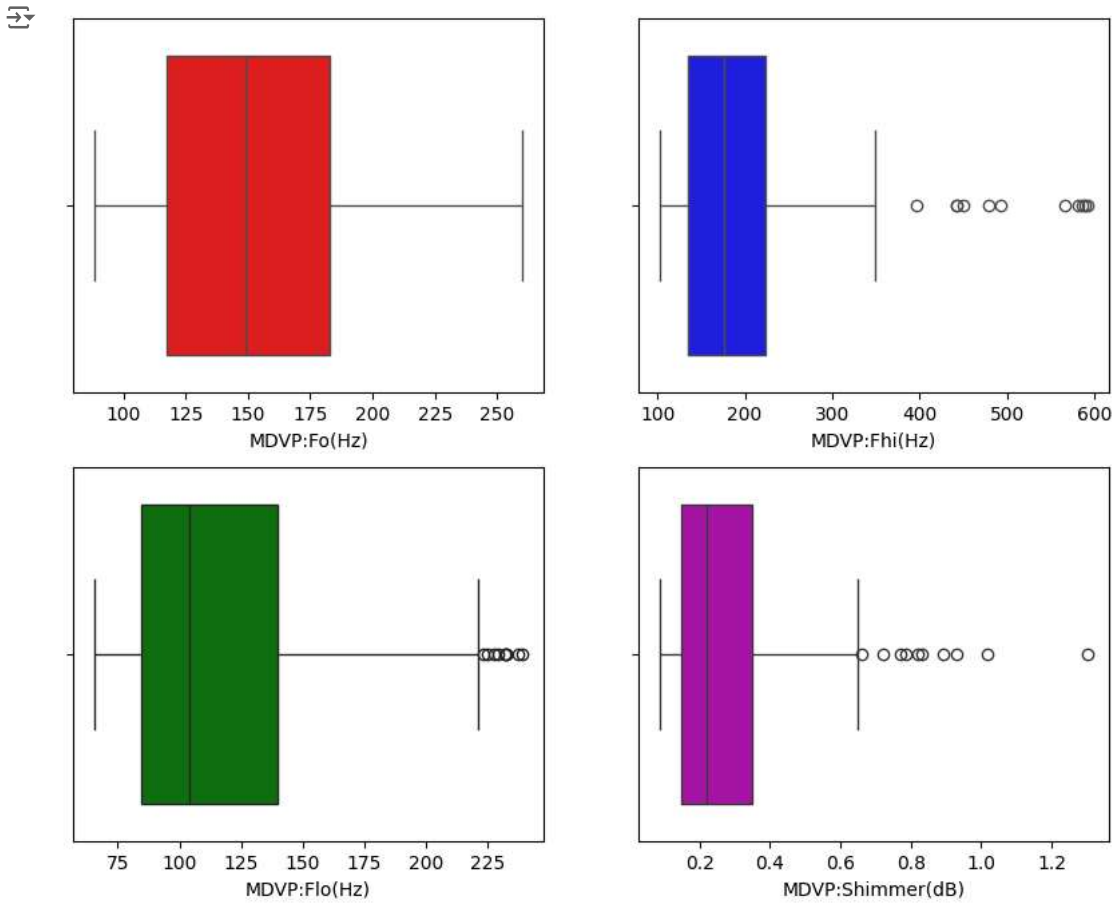
```
Empty DataFrame
Columns: [MDVP:Fo(Hz), MDVP:Fhi(Hz), MDVP:Flo(Hz), MDVP:Jitter(%), MDVP:Jitter(Abs), MDVP:RAP, MDVP:PPQ, Jitter:DDP, MDVP:Shimmer, MDVP:
Index: []

[0 rows x 23 columns]
Data without outliers (Z-score):
0      False
```

```
1      False
2      False
3      False
4      False
       ...
190    False
191    False
192    False
193    False
194    False
Length: 195, dtype: bool
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats


df = pd.DataFrame(disease)

# Visualize data before removing outliers using box plots
plt.figure(figsize=(5,5))
plt.suptitle('Box Plots Before Removing Outliers', fontsize=16)

for i, col in enumerate(['MDVP:Fhi(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Shimmer(dB)']):


    plt.subplot(2, 2, i+1)
    sns.boxplot(y=df[col])
    plt.title(f'{col}')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# Detect and remove outliers using Z-score
def remove_outliers(disease):

    Q1 = disease.quantile(0.25)
    Q3 = disease.quantile(0.75)
    IQR = Q3 - Q1


    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR


    df_no_outliers = disease[~((disease < lower_bound) | (disease > upper_bound)).any(axis=1)]
    return df_no_outliers

# Call the function to remove outliers and create df_no_outliers
df_no_outliers = remove_outliers(disease) # Call the function here

# Visualize data after removing outliers using box plots
plt.figure(figsize=(5,5))
plt.suptitle('Box Plots After Removing Outliers', fontsize=16)

for i, col in enumerate(['MDVP:Fhi(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Shimmer(dB)']):
    plt.subplot(2, 2, i+1)
    sns.boxplot(y=df_no_outliers[col])
    plt.title(f'{col}')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```




```
1 fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(10, 8))
2 sns.boxplot(disease, x = 'MDVP:Fo(Hz)', ax = ax1, color = 'r')
3 sns.boxplot(disease, x = 'MDVP:Fhi(Hz)', ax = ax2, color = 'b')
4 sns.boxplot(disease, x = 'MDVP:Flo(Hz)', ax = ax3, color = 'g')
5 sns.boxplot(disease, x = 'MDVP:Shimmer(dB)', ax = ax4, color = 'm')
6 plt.show()
```




Double-click (or enter) to edit

Double-click (or enter) to edit

```
1
2
3 z_scores = stats.zscore(disease)
```

```
1 print(z_scores)
```

```
      MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0       -0.829300     -0.436165     -0.952037        0.334914
1       -0.770972     -0.530974     -0.057721        0.715418
2       -0.909476     -0.723168     -0.109875        0.884991
3       -0.909622     -0.649092     -0.114229        0.775389
4       -0.925657     -0.606245     -0.130608        1.368893
..            ...           ...           ...             ...
190      0.483467      0.371185     -0.508265       -0.337173
191      1.339202      0.612690     -0.618218       -0.120037
192      0.495578      0.470104     -0.968393        1.526058
193      1.078761      2.190044     -0.954180        0.243924
194      1.454817      0.692246     -0.883481       -0.113833

      MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
0             0.749759  0.132963  0.760800    0.131755      0.745985
1             1.037674  0.453892  1.276809    0.452684      1.681731
2             1.325589  0.720770  1.585687    0.721813      1.202693
3             1.325589  0.578885  1.284076    0.577677      1.340396
4             1.901418  1.095750  2.047187    1.096793      1.836448
..                 ...       ...       ...         ...           ...
190          -0.401899 -0.228505 -0.311189   -0.227459      0.593395
191          -0.401899  0.001213 -0.191272    0.002258     -0.116922
192           1.037674  0.991026  0.797139    0.992069     -0.352453
193          -0.113985  0.132963  0.164847    0.131755     -0.358834
194          -0.401899 -0.120403 -0.100425   -0.120483     -0.577883

      MDVP:Shimmer(dB)  ...  Shimmer:DDA       NHR       HNR    status  \
0             0.739536  ...     0.607532 -0.067893 -0.193225  0.571429
1             1.768464  ...     1.548254 -0.137843 -0.634508  0.571429
```

```
2           1.027636  ...     1.175323 -0.291633 -0.279760  0.571429
3           1.207698  ...     1.340229 -0.280719 -0.281346  0.571429
4           1.552389  ...     1.899461 -0.178026 -0.506745  0.571429
..               ...  ...          ...       ...       ...       ...
190         0.631498  ...     0.759930  0.069278 -0.536647 -1.750000
191        -0.099041  ...     0.037108 -0.167360 -0.620463 -1.750000
192        -0.135053  ...    -0.294679  2.041513 -0.906799 -1.750000
193        -0.212223  ...    -0.297970  1.175327 -0.649233 -1.750000
194        -0.474600  ...    -0.533645  0.474590 -0.153356 -1.750000

         RPDE       DFA   spread1   spread2        D2       PPE
0   -0.807838  1.760814  0.801323  0.480477 -0.210531  0.868886
1   -0.387524  1.837562  1.479853  1.311185  0.275077  1.803605
2   -0.662075  1.942048  1.141445  1.017682 -0.103629  1.402661
3   -0.613134  1.832380  1.440945  1.293840  0.062145  1.806954
4   -0.783021  1.909364  1.780940  0.096195 -0.130026  2.267082
..        ...       ...       ...       ...       ...       ...
190 -0.483208 -1.090704 -0.785527 -1.256837  0.721944 -0.817703
191 -0.644916 -0.631503 -0.469859 -1.168475  1.054135 -0.418929
192 -0.877441 -1.130853 -1.014154 -0.818079  0.780338 -0.832410
193 -0.456374 -1.343323 -0.974960 -0.229066 -0.637003 -0.926105
194 -0.344660 -0.973698 -0.036471 -0.430853  0.454802 -0.645055

[195 rows x 23 columns]
```

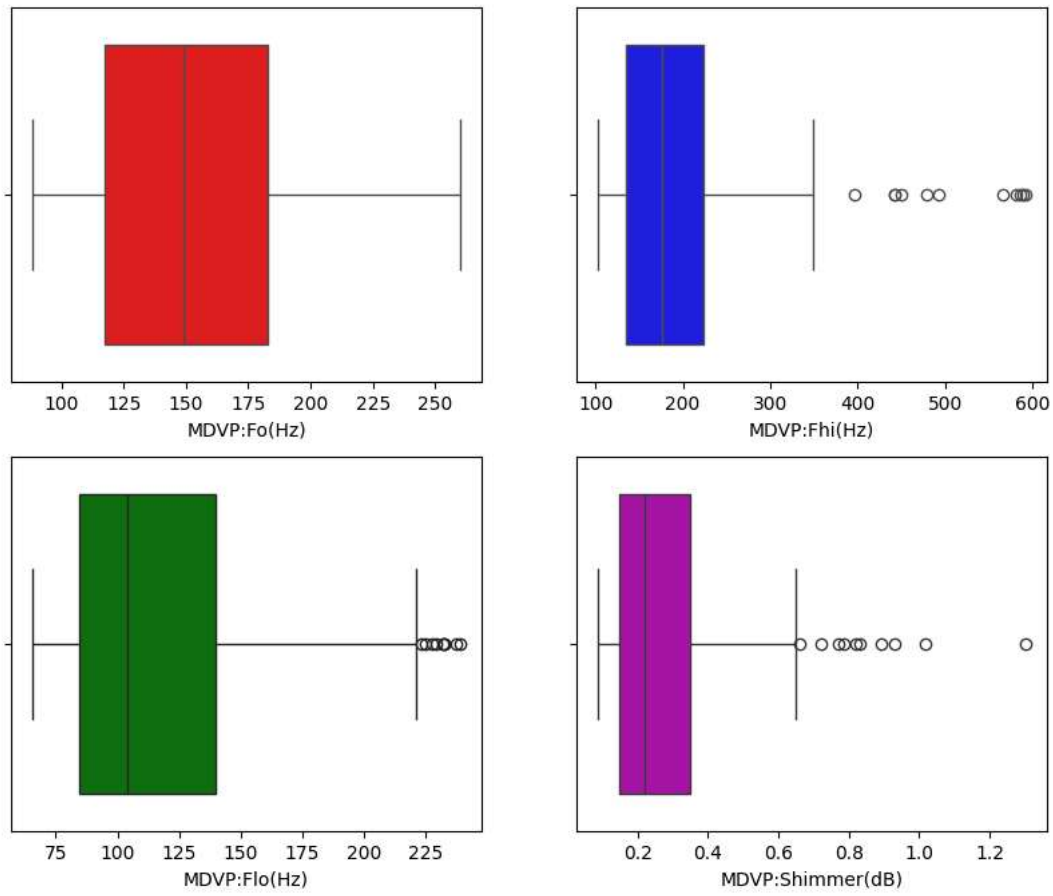
```
def remove_outliers(disease):

    Q1 = disease.quantile(0.25)
    Q3 = disease.quantile(0.75)
    IQR = Q3 - Q1


    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR


    df_no_outliers = disease[~((disease < lower_bound) | (disease > upper_bound)).any(axis=1)]

    return df_no_outliers
```

```
data_no_outliers = remove_outliers(disease)

print("Original Data:")
print(disease)

print("\nData after removing outliers:")
print(data_no_outliers)
```

```
Original Data:
     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)  ...  Shimmer:DDA      NHR     HNR  status      RPDE  \
0               0.426  ...      0.06545  0.02211  21.033       1  0.414783
1               0.626  ...      0.09403  0.01929  19.085       1  0.458359
2               0.482  ...      0.08270  0.01309  20.651       1  0.429895
3               0.517  ...      0.08771  0.01353  20.644       1  0.434969
4               0.584  ...      0.10470  0.01767  19.649       1  0.417356
```

```
..              ...  ...         ...      ...     ...     ...       ...
190            0.405  ...     0.07008  0.02764  19.517       0  0.448439
191            0.263  ...     0.04812  0.01810  19.147       0  0.431674
192            0.256  ...     0.03804  0.10715  17.883       0  0.407567
193            0.241  ...     0.03794  0.07223  19.020       0  0.451221
194            0.190  ...     0.03078  0.04398  21.209       0  0.462803

          DFA   spread1   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 23 columns]

Data after removing outliers:
     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0        119.992       157.302        74.997         0.00784
```

```
1 fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(10, 8))
2 sns.boxplot(disease, x = 'MDVP:Fo(Hz)', ax = ax1, color = 'r')
3 sns.boxplot(disease, x = 'MDVP:Fhi(Hz)', ax = ax2, color = 'b')
4 sns.boxplot(disease, x = 'MDVP:Flo(Hz)', ax = ax3, color = 'g')
5 sns.boxplot(disease, x = 'MDVP:Shimmer(dB)', ax = ax4, color = 'm')
6 plt.show()
```




Double-click (or enter) to edit

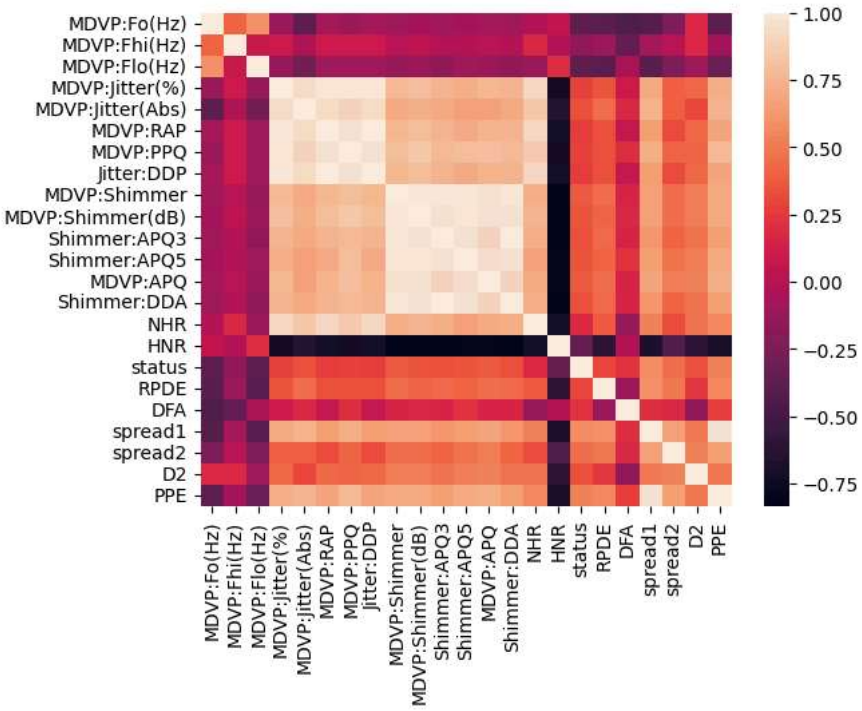
Double-click (or enter) to edit

```
1 disease['status'].value_counts()
```

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

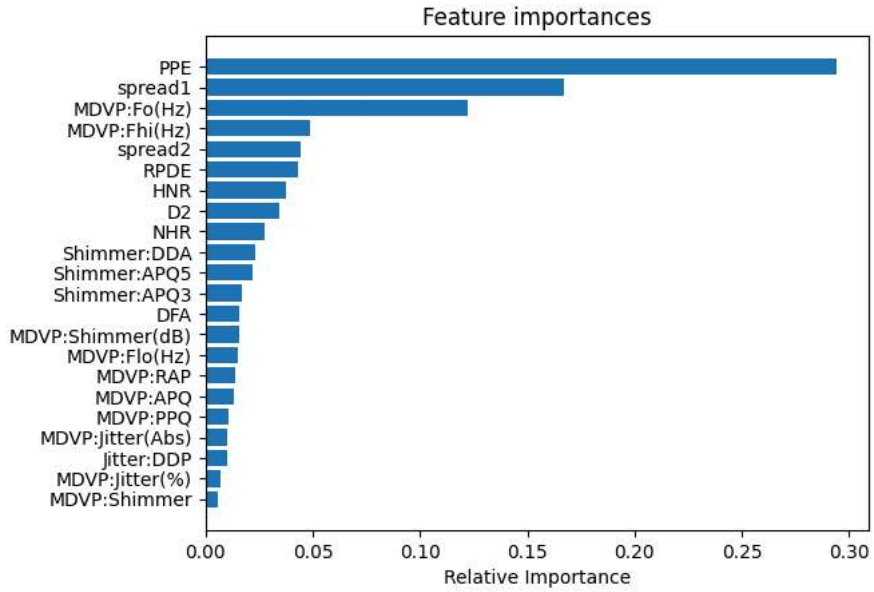
```
sns.heatmap(disease.corr())
```

```
<Axes: >
```




```
x = disease.drop(columns=['status'])
y = disease['status']
```

```
from sklearn.ensemble import RandomForestRegressor


rf = RandomForestRegressor()
rf.fit(x, y)


importances = rf.feature_importances_
indices = np.argsort(importances)

plt.figure()
plt.title("Feature importances")
plt.barh(range(x.shape[1]), importances[indices], align='center')
plt.yticks(range(x.shape[1]), [x.columns[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()

```




```
1 Start coding or generate with AI.
```

```
1 from sklearn.preprocessing import StandardScaler
2 from sklearn.model_selection import train_test_split
3 from sklearn import svm
4 from sklearn.metrics import accuracy_score
```

```
1 scaler = StandardScaler()
2 x_std = scaler.fit_transform(x)
```

```
1 x_std
```

```
array([[-0.82929965, -0.43616456, -0.95203729, ...,  0.48047686,
        -0.21053082,  0.86888575],
       [-0.77097169, -0.53097409, -0.05772056, ...,  1.31118546,
         0.27507712,  1.80360503],
       [-0.90947638, -0.7231683 , -0.10987483, ...,  1.01768236,
        -0.10362861,  1.40266141],
       ...,
       [ 0.49557839,  0.47010361, -0.96839309, ..., -0.81807931,
         0.78033848, -0.83241014],
       [ 1.07876114,  2.19004398, -0.95417967, ..., -0.22906571,
        -0.63700298, -0.92610456],
       [ 1.45481664,  0.69224632, -0.88348115, ..., -0.43085284,
         0.45480231, -0.64505466]])
```

```
1 x_train,x_test,y_train,y_test = train_test_split(x_std,y,test_size = 0.2 , stratify=y , random_state=2)
```

```
1 x_train.shape , x_test.shape
```

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

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

```
1 model.fit(x_train,y_train)
```

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

```
1 pred_train = model.predict(x_train)
2 accuracy_score(pred_train,y_train)
```

```
0.8910256410256411
```

```
pred_test = model.predict(x_test)
accuracy_score(pred_test,y_test)
```

```
0.8974358974358975
```

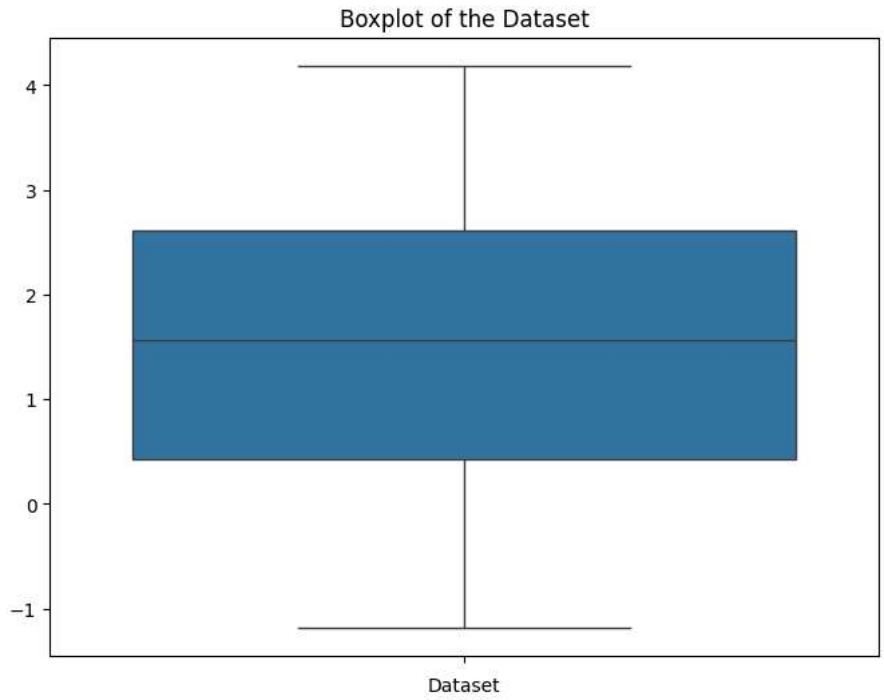
```
input_data = (223.365,238.987,98.664,0.00264,0.00001,0.00154,0.00151,0.00461,0.01906,0.165,0.01013,0.01296,0.0134,0.03039,0.00301,26.138,


input_data_array = np.asarray(input_data)


input_data_reshape = input_data_array.reshape(1,-1)


std_data = scaler.transform(input_data_reshape,)


prediction = model.predict(std_data)

print(prediction)
if prediction == 0:
    print("The Person does not have Parkinsson's Disease")
else:
    print("The Person is suffering from parkinsson's Disease")
```

```
[0]
The Person does not have Parkinsson's Disease
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fi
  warnings.warn(
```

```
from scipy.stats import skew


skewness = skew(disease)
print(f"Skewness: {skewness}")

```

```
Skewness: [ 0.58717588  2.52254909  1.20796613  3.06116496  2.62865025  3.33480141
  3.05019642  3.336141    1.65363384  1.98397574  1.56839203  1.78483126
  2.5978645   1.56843332  4.18817251 -0.51035273 -1.17857143 -0.14229695
 -0.03295762  0.42880766  0.1433171   0.42706615  0.79134337]
```

```


plt.figure(figsize=(8, 6))
sns.boxplot(data=skewness)
plt.title('Boxplot of the Dataset')
plt.xlabel('Dataset')


plt.show()

```



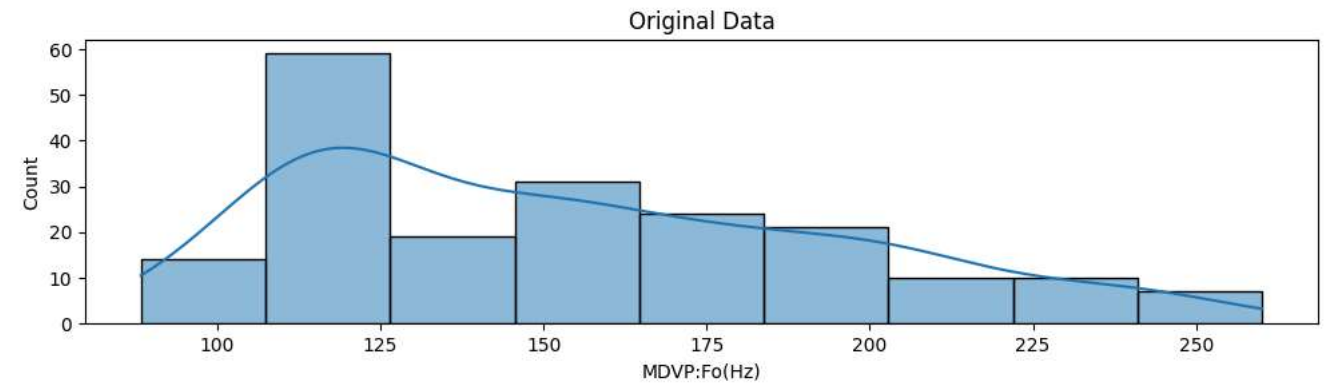

```
1 disease.columns
```

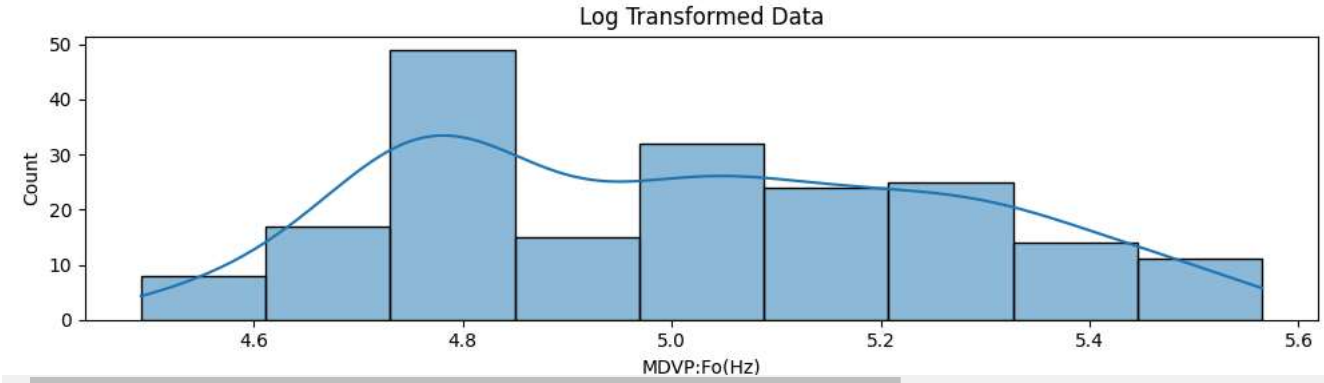
```
Index(['MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

```
1 Start coding or generate with AI.
```

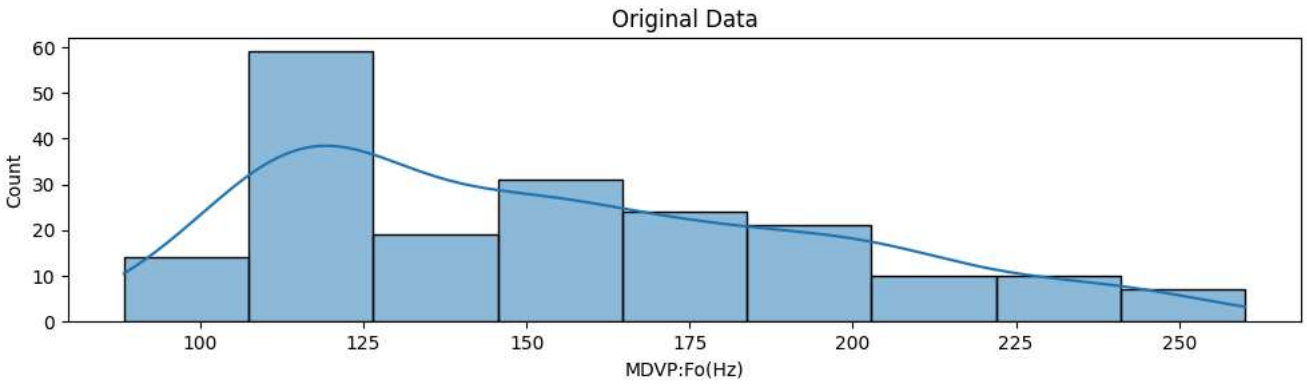
```
1
2
3
```

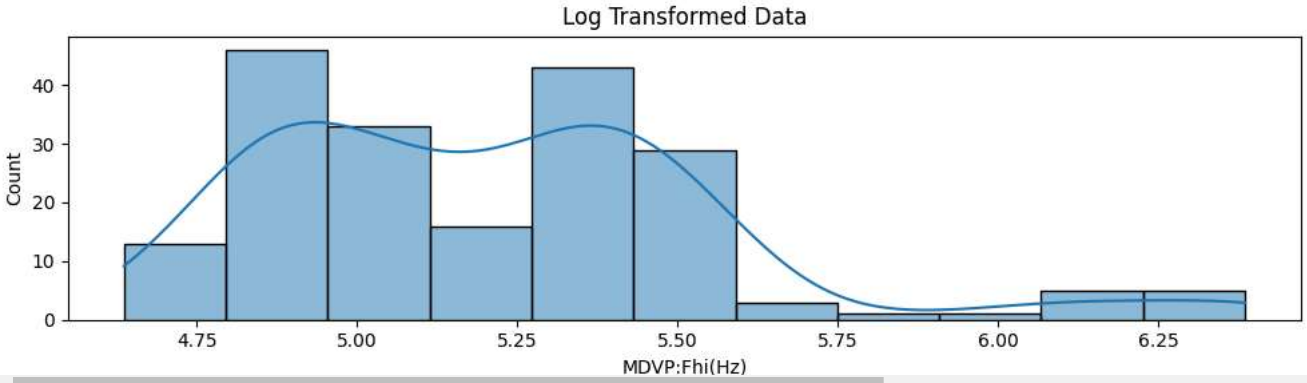
```
1
2 if len(disease) > 10000:
3     disease_sampled = disease.sample(10000)
4 else:
5     disease_sampled = disease
6 log_data = np.log1p(disease_sampled['MDVP:Fo(Hz)'])
7 plt.figure(figsize=(10, 6))
8 plt.subplot(2, 1, 1)
9 sns.histplot(disease_sampled['MDVP:Fo(Hz)'], kde=True)
10 plt.title('Original Data')
11 plt.subplot(2, 1, 2)
12 sns.histplot(log_data, kde=True)
13 plt.title('Log Transformed Data')
14
15 plt.tight_layout()
16 plt.show()
17
```



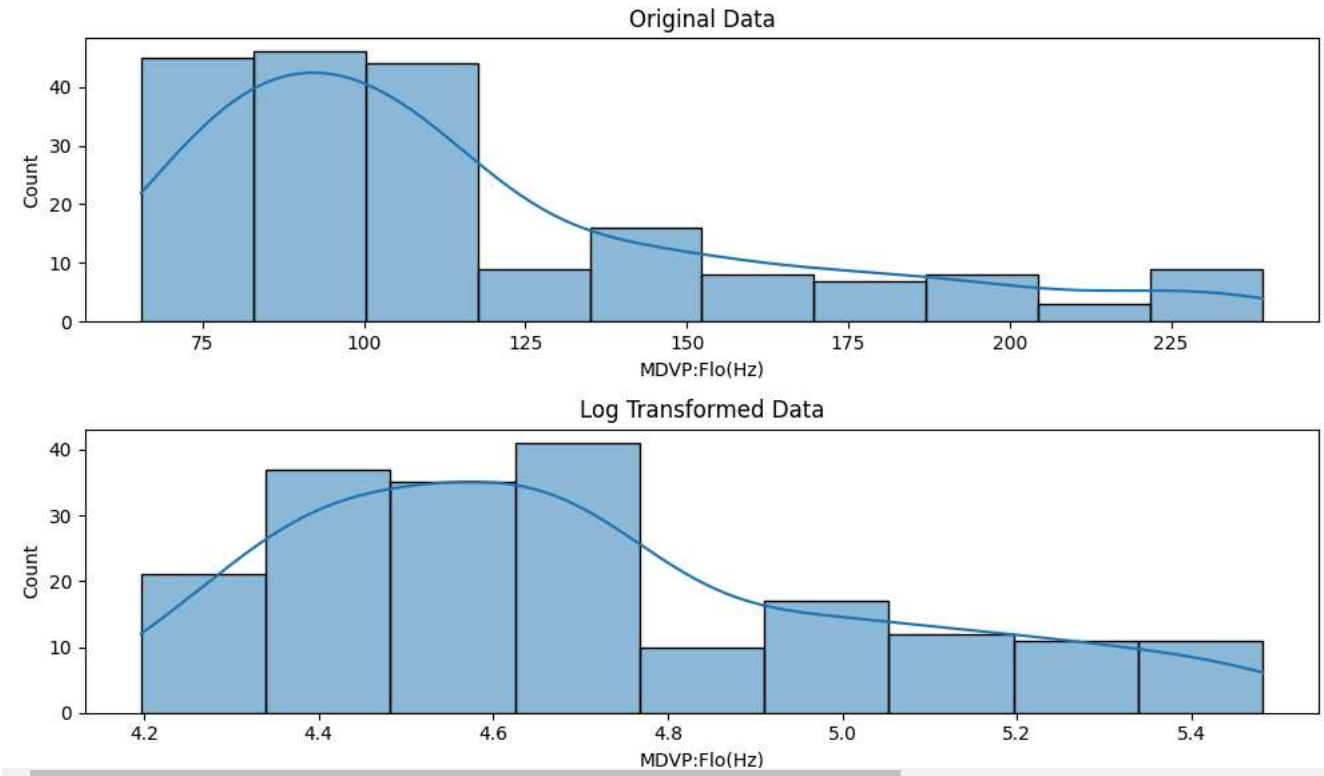




```
1
2
3
4 if len(disease) > 10000:
5     disease_sampled = disease.sample(10000)
6 else:
7     disease_sampled = disease
8 log_data = np.log1p(disease_sampled['MDVP:Fhi(Hz)'])
9 plt.figure(figsize=(10, 6))
10 plt.subplot(2, 1, 1)
11 sns.histplot(disease_sampled['MDVP:Fo(Hz)'], kde=True)
12 plt.title('Original Data')
13 plt.subplot(2, 1, 2)
14 sns.histplot(log_data, kde=True)
15 plt.title('Log Transformed Data')
16
17 plt.tight_layout()
18 plt.show()
```

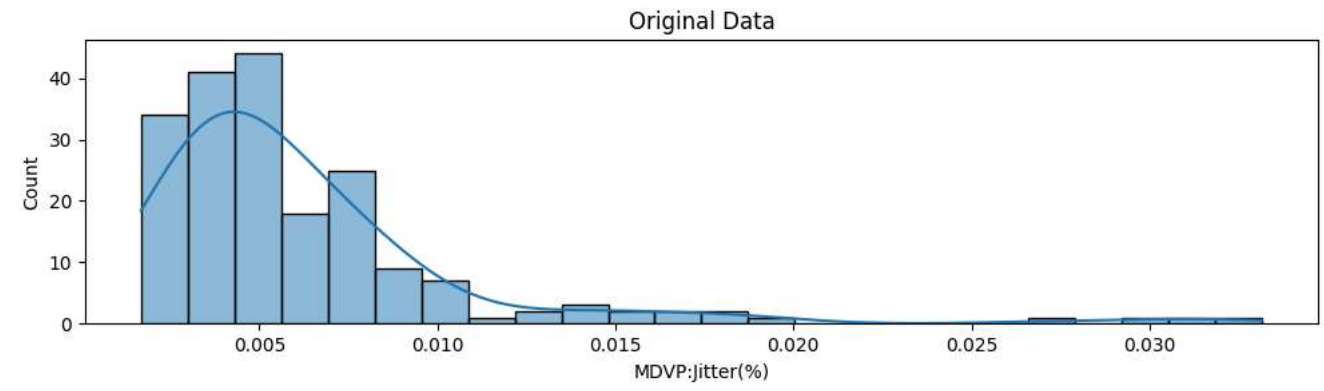


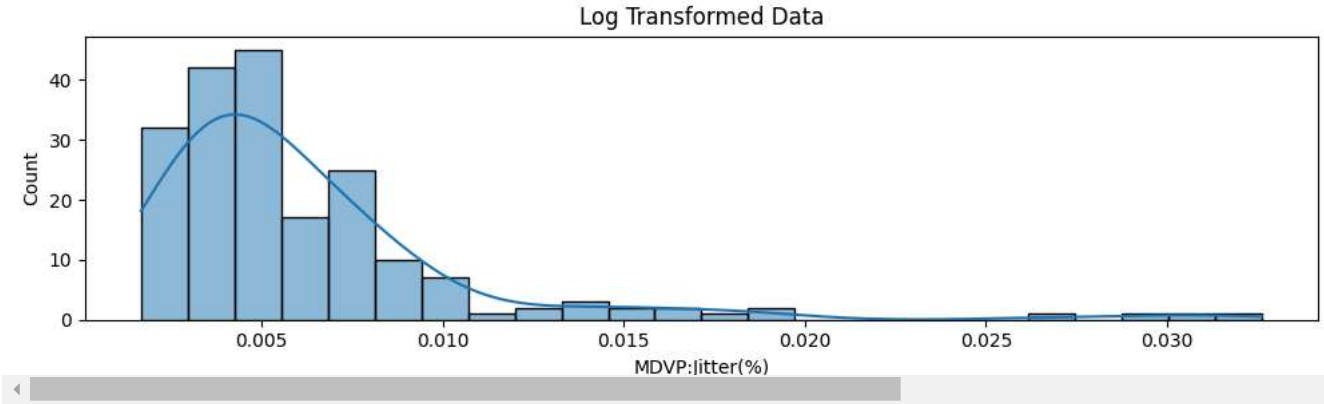




```
1
2
3 if len(disease) > 10000:
4     disease_sampled = disease.sample(10000)
5 else:
6     disease_sampled = disease
7 log_data = np.log1p(disease_sampled['MDVP:Flo(Hz)'])
8 plt.figure(figsize=(10, 6))
9 plt.subplot(2, 1, 1)
10 sns.histplot(disease_sampled['MDVP:Flo(Hz)'], kde=True)
11 plt.title('Original Data')
12 plt.subplot(2, 1, 2)
13 sns.histplot(log_data, kde=True)
14 plt.title('Log Transformed Data')
15
16 plt.tight_layout()
17 plt.show()
18
```

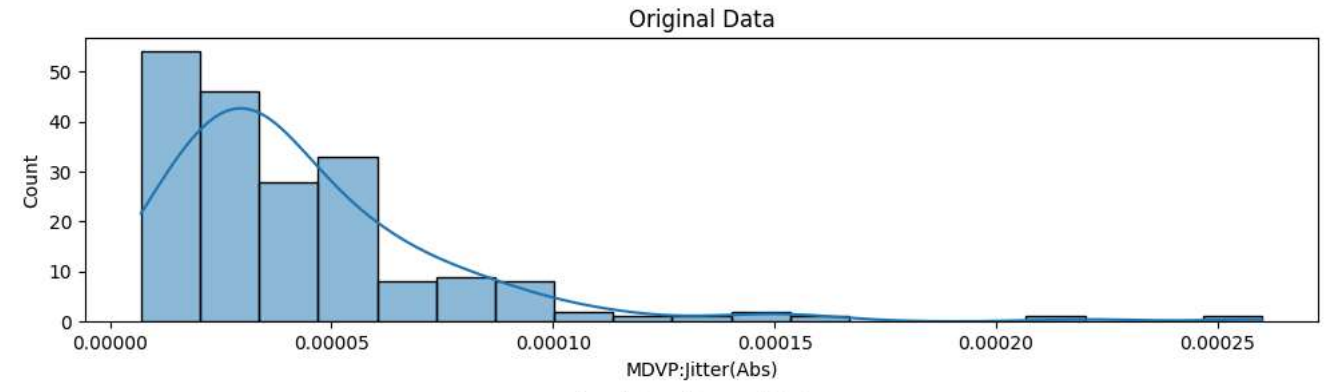


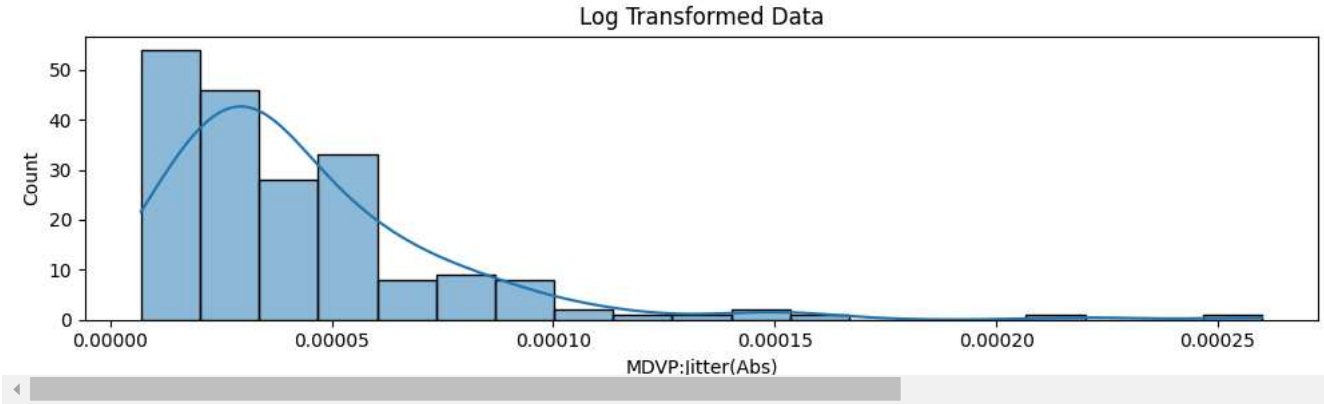

```
1
2 if len(disease) > 10000:
3     disease_sampled = disease.sample(10000)
4 else:
5     disease_sampled = disease
6 log_data = np.log1p(disease_sampled['MDVP:Jitter(%)'])
7 plt.figure(figsize=(10, 6))
8 plt.subplot(2, 1, 1)
9 sns.histplot(disease_sampled['MDVP:Jitter(%)'], kde=True)
10 plt.title('Original Data')
11 plt.subplot(2, 1, 2)
12 sns.histplot(log_data, kde=True)
13 plt.title('Log Transformed Data')
14
15 plt.tight_layout()
16 plt.show()
```




```
1
2 if len(disease) > 10000:
3     disease_sampled = disease.sample(10000)
4 else:
5     disease_sampled = disease
6 log_data = np.log1p(disease_sampled['MDVP:Jitter(Abs)'])
7 plt.figure(figsize=(10, 6))
8 plt.subplot(2, 1, 1)
9 sns.histplot(disease_sampled['MDVP:Jitter(Abs)'], kde=True)
10 plt.title('Original Data')
11 plt.subplot(2, 1, 2)
12 sns.histplot(log_data, kde=True)
13 plt.title('Log Transformed Data')
14
15 plt.tight_layout()
16 plt.show()
```

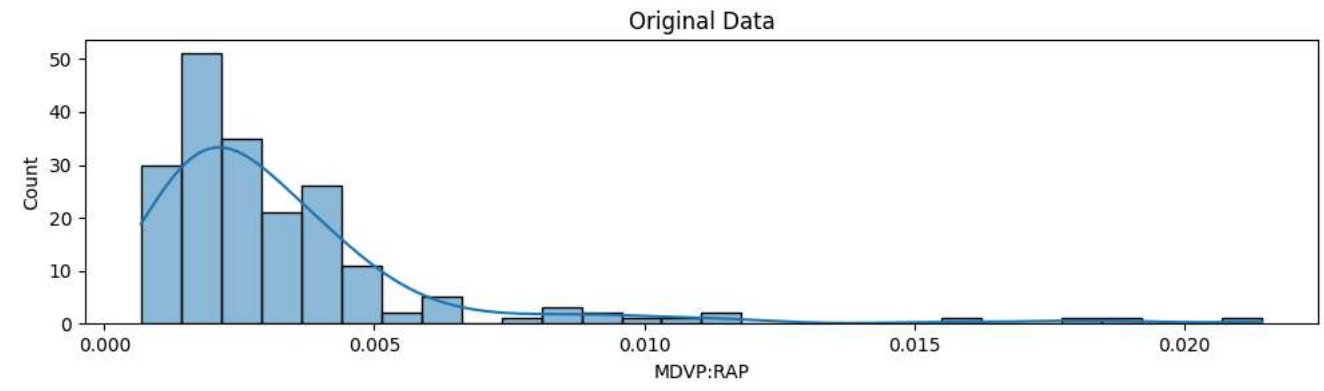


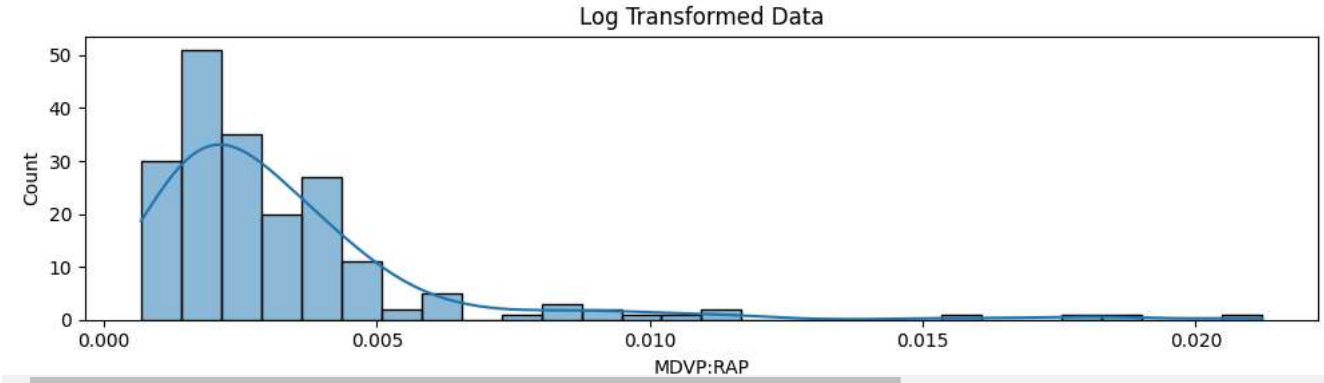




```
1 disease.columns
```

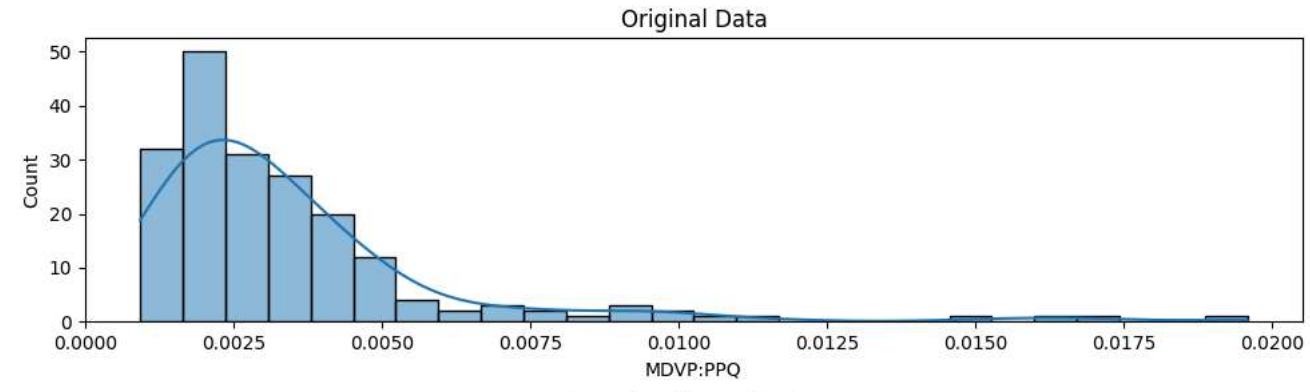
```
Index(['MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

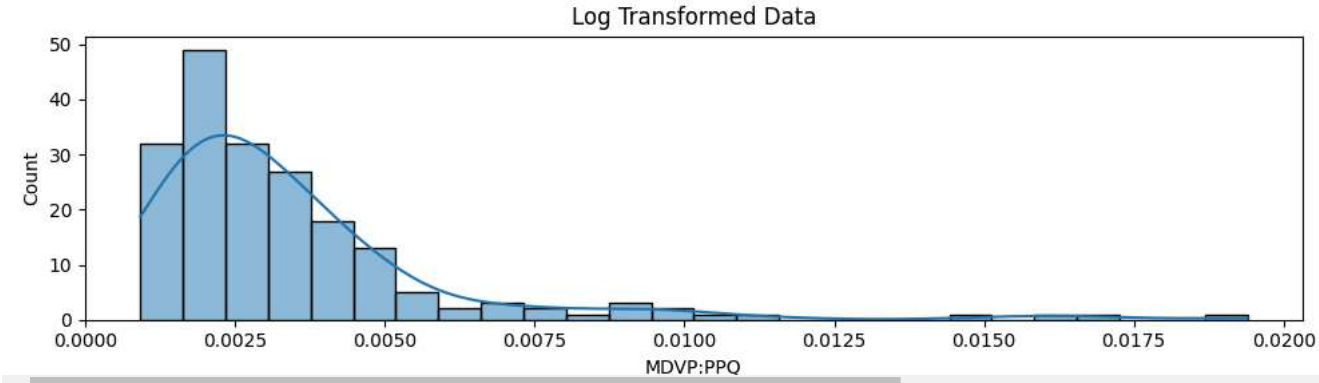
```
1 if len(disease) > 10000:
2     disease_sampled = disease.sample(10000)
3 else:
4     disease_sampled = disease
5 log_data = np.log1p(disease_sampled['MDVP:RAP'])
6 plt.figure(figsize=(10, 6))
7 plt.subplot(2, 1, 1)
8 sns.histplot(disease_sampled['MDVP:RAP'], kde=True)
9 plt.title('Original Data')
10 plt.subplot(2, 1, 2)
11 sns.histplot(log_data, kde=True)
12 plt.title('Log Transformed Data')
13
14 plt.tight_layout()
15 plt.show()
```




```
if len(disease) > 10000:
    disease_sampled = disease.sample(10000)
else:
    disease_sampled = disease
log_data = np.log1p(disease_sampled['MDVP:PPQ'])
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
sns.histplot(disease_sampled['MDVP:PPQ'], kde=True)
plt.title('Original Data')
plt.subplot(2, 1, 2)
sns.histplot(log_data, kde=True)
plt.title('Log Transformed Data')

plt.tight_layout()
plt.show()
```

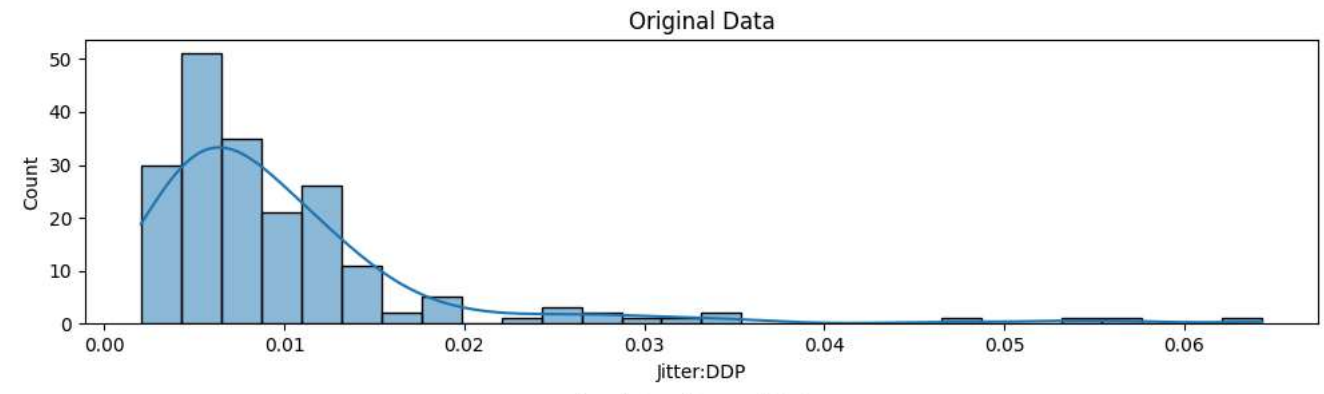


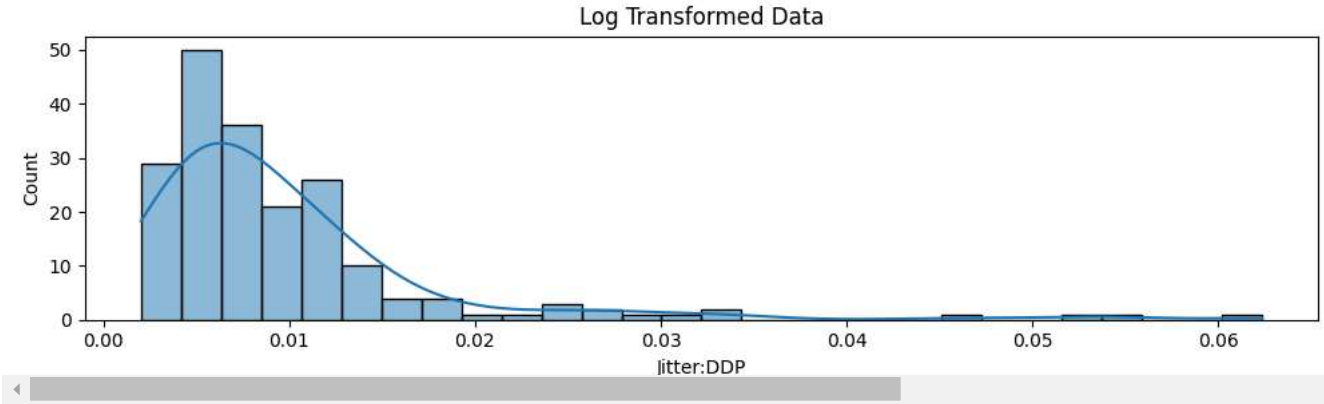




```
1 disease.columns
```

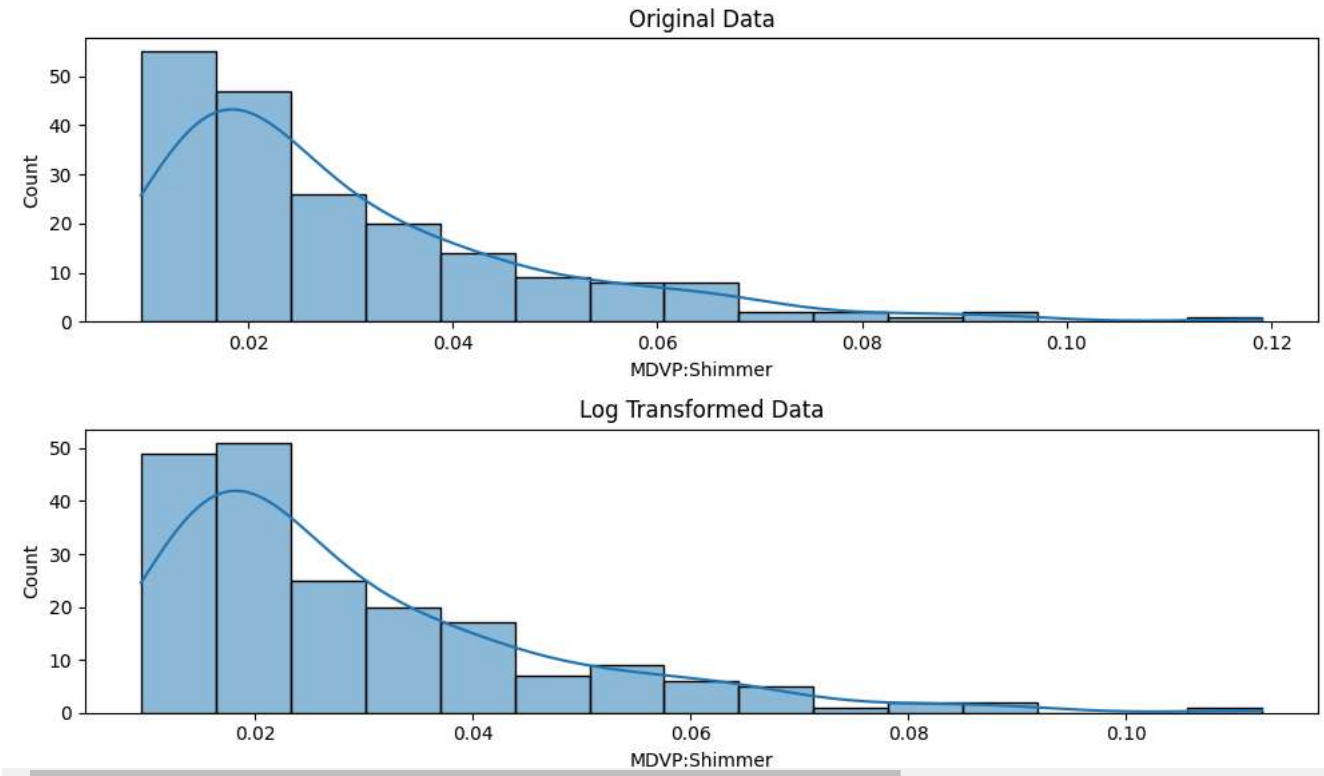
```
Index(['MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

```
1 if len(disease) > 10000:
2     disease_sampled = disease.sample(10000)
3 else:
4     disease_sampled = disease
5 log_data = np.log1p(disease_sampled['Jitter:DDP'])
6 plt.figure(figsize=(10, 6))
7 plt.subplot(2, 1, 1)
8 sns.histplot(disease_sampled['Jitter:DDP'], kde=True)
9 plt.title('Original Data')
10 plt.subplot(2, 1, 2)
11 sns.histplot(log_data, kde=True)
12 plt.title('Log Transformed Data')
13
14 plt.tight_layout()
15 plt.show()
```

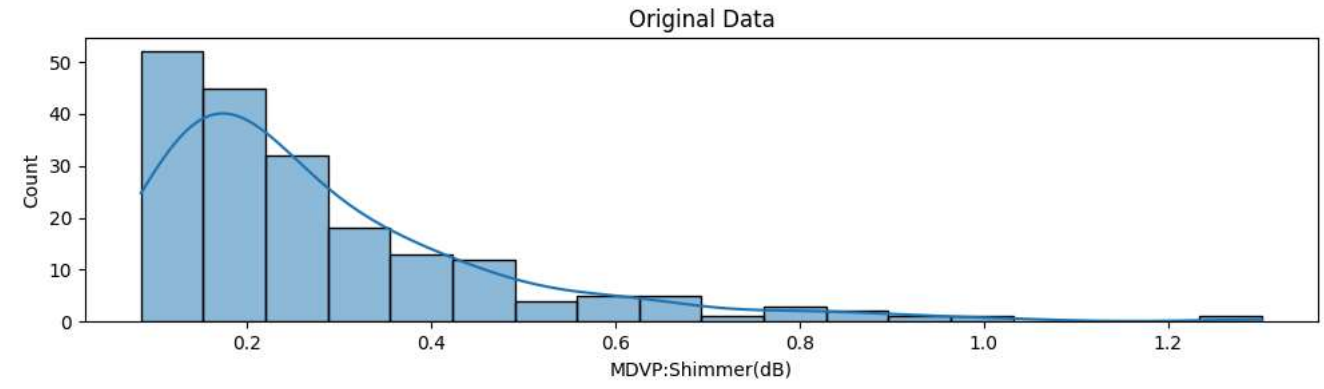


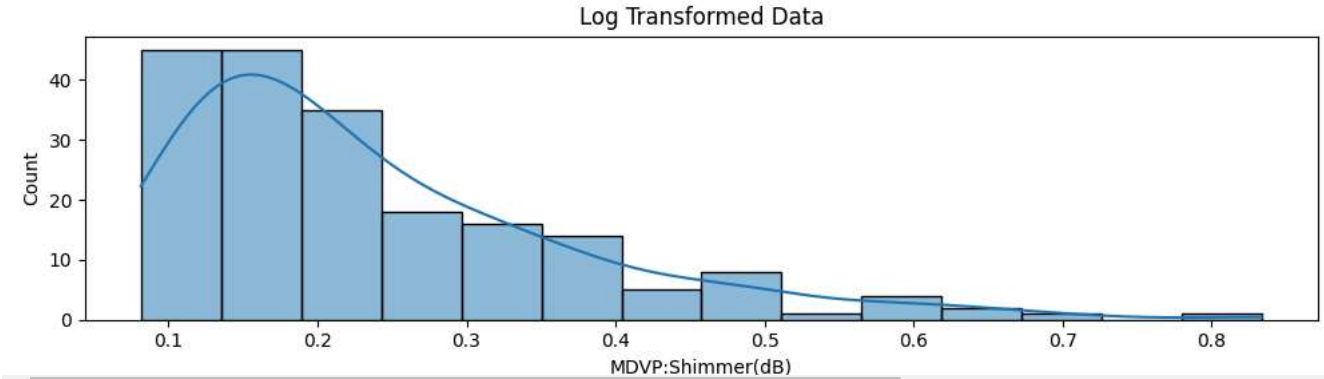




```
if len(disease) > 10000:
    disease_sampled = disease.sample(10000)
else:
    disease_sampled = disease
log_data = np.log1p(disease_sampled['MDVP:Shimmer'])
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
sns.histplot(disease_sampled['MDVP:Shimmer'], kde=True)
plt.title('Original Data')
plt.subplot(2, 1, 2)
sns.histplot(log_data, kde=True)
plt.title('Log Transformed Data')

plt.tight_layout()
plt.show()
```

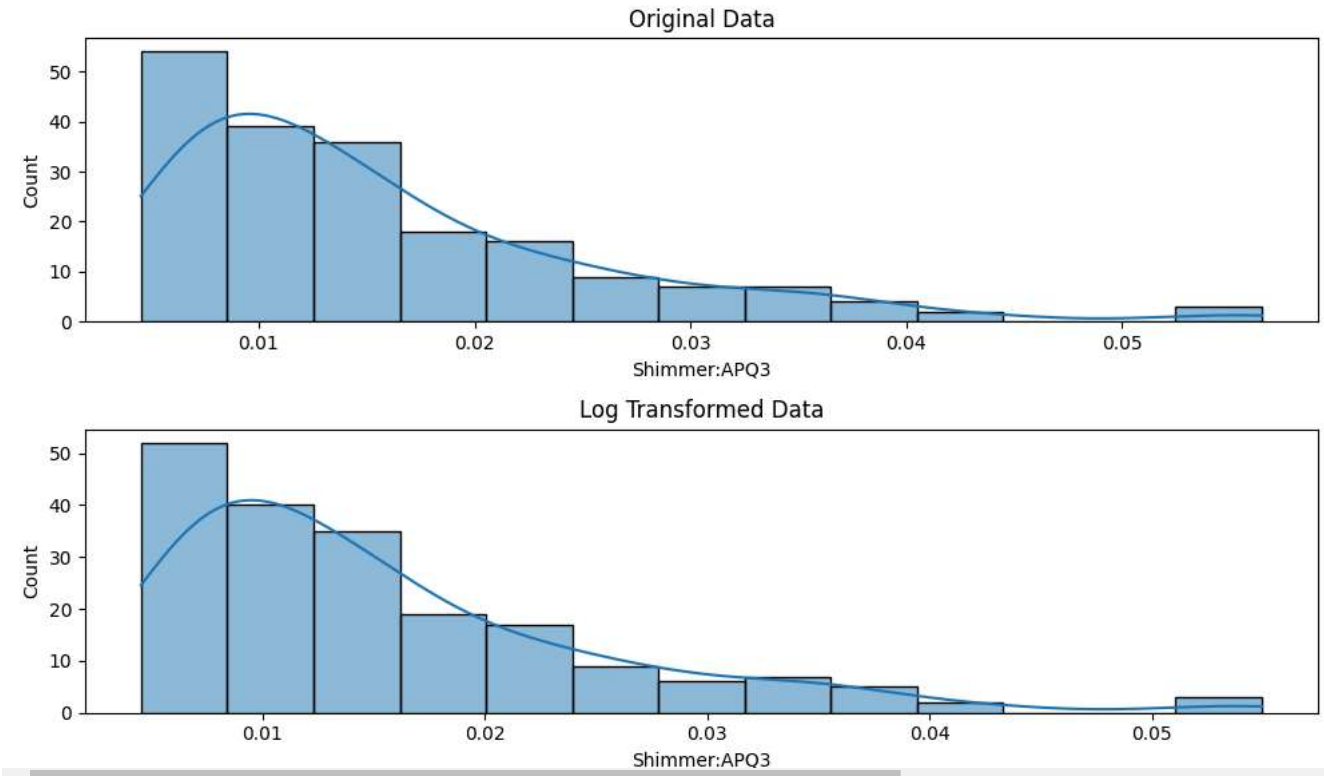



```
if len(disease) > 10000:
    disease_sampled = disease.sample(10000)
else:
    disease_sampled = disease
log_data = np.log1p(disease_sampled['MDVP:Shimmer(dB)'])
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
sns.histplot(disease_sampled['MDVP:Shimmer(dB)'], kde=True)
plt.title('Original Data')
plt.subplot(2, 1, 2)
sns.histplot(log_data, kde=True)
plt.title('Log Transformed Data')

plt.tight_layout()
plt.show()
```

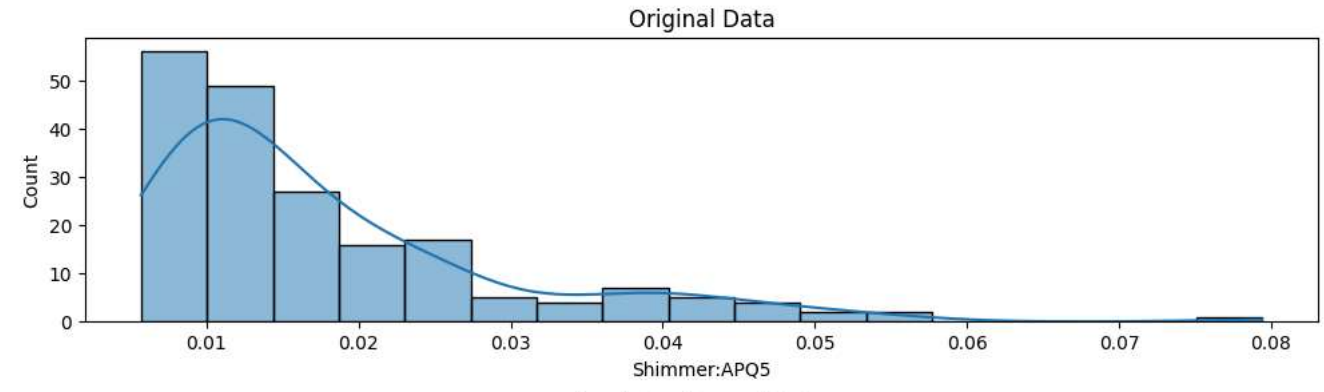


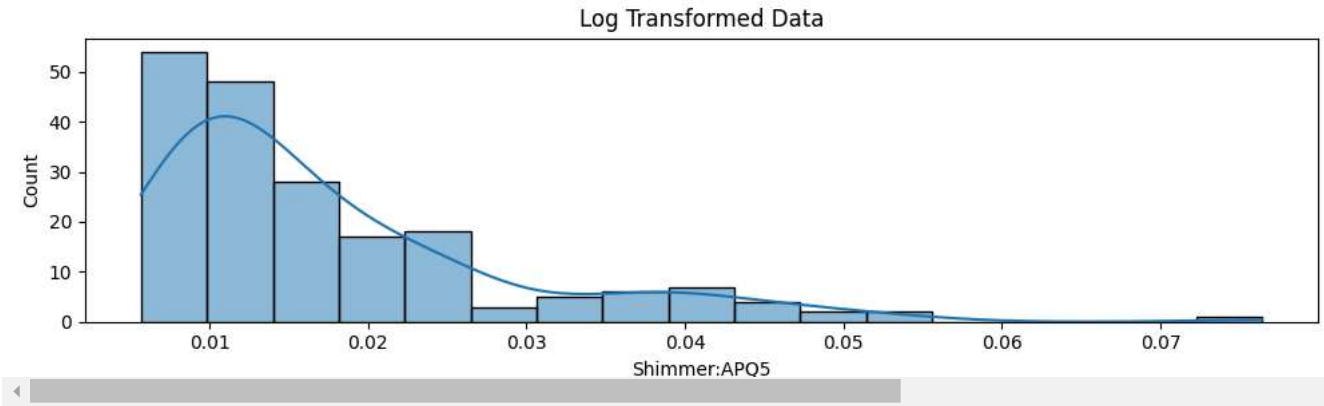




```
if len(disease) > 10000:
    disease_sampled = disease.sample(10000)
else:
    disease_sampled = disease
log_data = np.log1p(disease_sampled['Shimmer:APQ3'])
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
sns.histplot(disease_sampled['Shimmer:APQ3'], kde=True)
plt.title('Original Data')
plt.subplot(2, 1, 2)
sns.histplot(log_data, kde=True)
plt.title('Log Transformed Data')

plt.tight_layout()
plt.show()
```

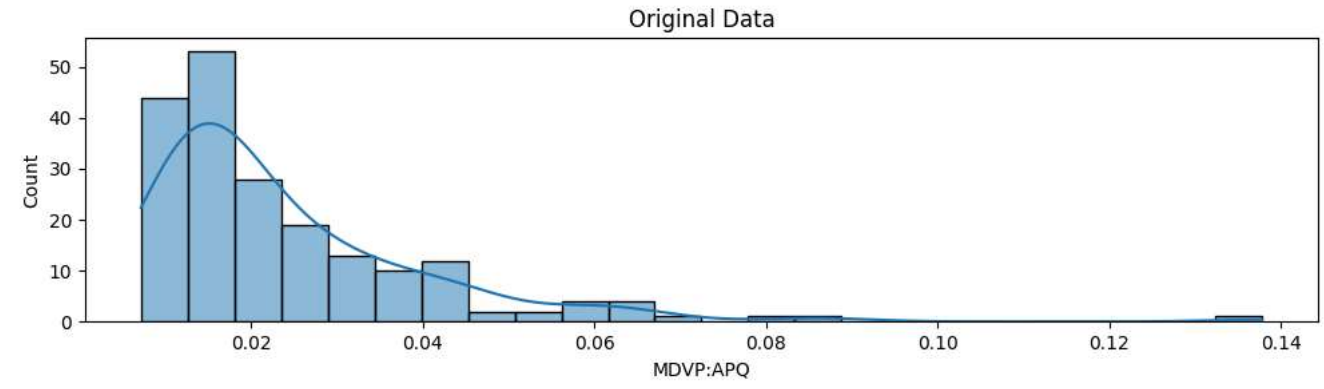


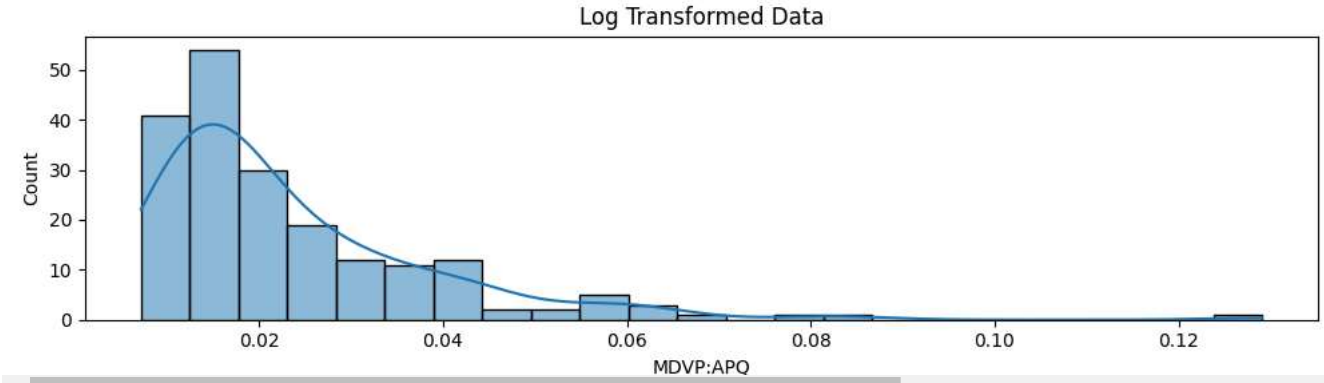

```
if len(disease) > 10000:
    disease_sampled = disease.sample(10000)
else:
    disease_sampled = disease
log_data = np.log1p(disease_sampled['Shimmer:APQ5'])
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
sns.histplot(disease_sampled['Shimmer:APQ5'], kde=True)
plt.title('Original Data')
plt.subplot(2, 1, 2)
sns.histplot(log_data, kde=True)
plt.title('Log Transformed Data')

plt.tight_layout()
plt.show()
```




```
if len(disease) > 10000:
    disease_sampled = disease.sample(10000)
else:
    disease_sampled = disease
log_data = np.log1p(disease_sampled['MDVP:APQ'])
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
sns.histplot(disease_sampled['MDVP:APQ'], kde=True)
plt.title('Original Data')
plt.subplot(2, 1, 2)
sns.histplot(log_data, kde=True)
plt.title('Log Transformed Data')

plt.tight_layout()
plt.show()
```

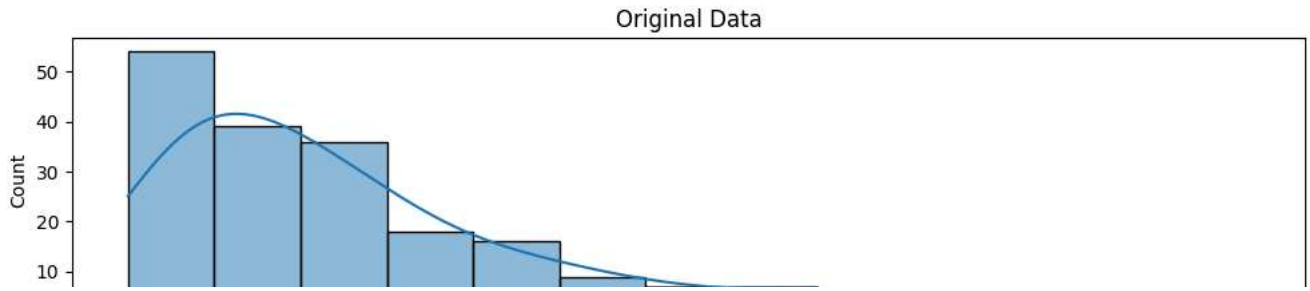


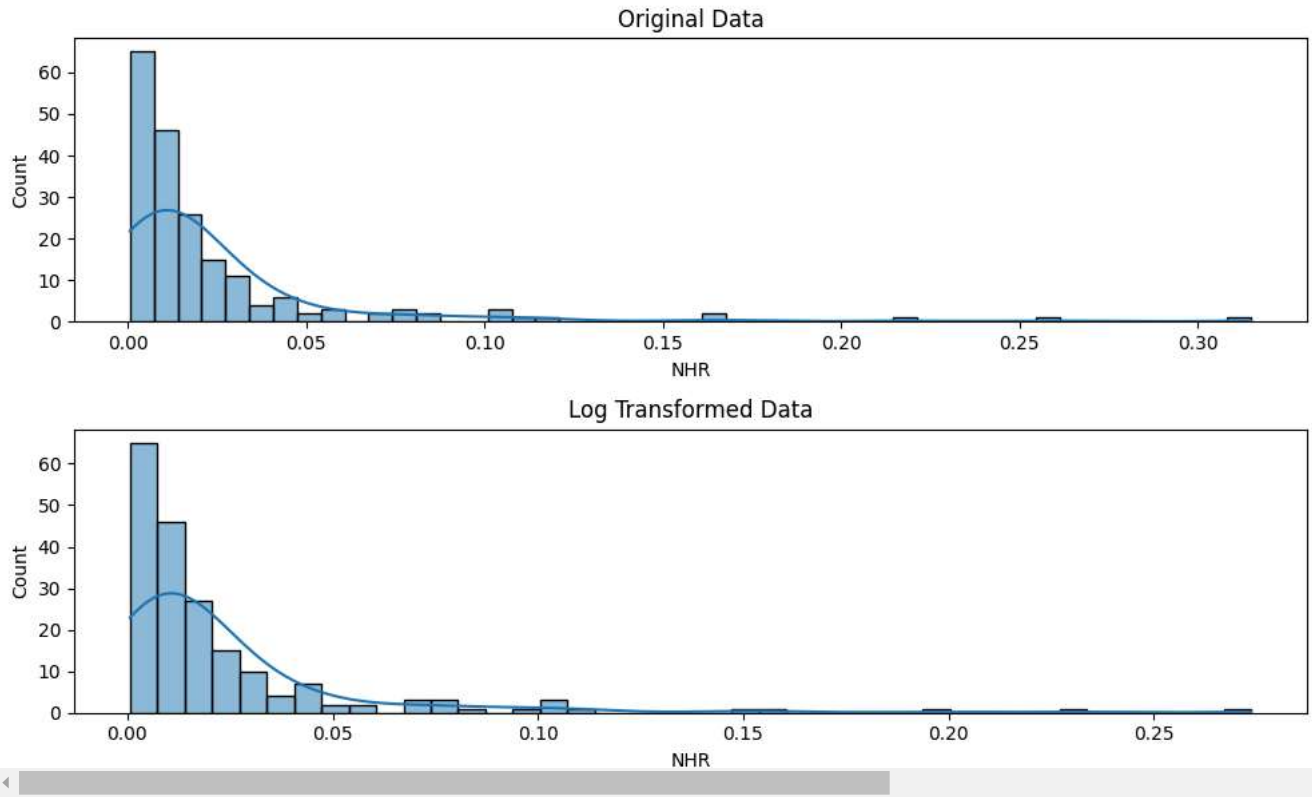

```
1 disease.columns
```

```
Index(['MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

```
1 if len(disease) > 10000:
2     disease_sampled = disease.sample(10000)
3 else:
4     disease_sampled = disease
5 log_data = np.log1p(disease_sampled['Shimmer:DDA'])
6 plt.figure(figsize=(10, 6))
7 plt.subplot(2, 1, 1)
8 sns.histplot(disease_sampled['Shimmer:DDA'], kde=True)
9 plt.title('Original Data')
10 plt.subplot(2, 1, 2)
11 sns.histplot(log_data, kde=True)
12 plt.title('Log Transformed Data')
13
14 plt.tight_layout()
15 plt.show()
```




```
if len(disease) > 10000:
    disease_sampled = disease.sample(10000)
else:
    disease_sampled = disease
log_data = np.log1p(disease_sampled['NHR'])
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
sns.histplot(disease_sampled['NHR'], kde=True)
plt.title('Original Data')
plt.subplot(2, 1, 2)
sns.histplot(log_data, kde=True)
plt.title('Log Transformed Data')

plt.tight_layout()
plt.show()
```




```
from scipy.stats import boxcox
data=disease['HNR']
```