ParkinsonsDisease_BusinessProject_ML

Importing the All neccesary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classificat
import os
```

Data set

In [2]:
```
df = pd.read_csv('Parkinsons.csv')
df.sample(5)
```

Out[2]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(A
188	phon_R01_S49_6	114.563	119.167	86.647	0.00327	0.00
43	phon_R01_S10_2	241.404	248.834	232.483	0.00281	0.00
157	phon_R01_S37_5	117.963	134.209	100.757	0.01813	0.00
122	phon_R01_S31_1	138.190	203.522	83.340	0.00704	0.00
132	phon_R01_S32_5	119.056	125.213	86.795	0.00346	0.00

5 rows × 24 columns

In [3]:
```
df.shape
```

Out[3]:
```
(195, 24)
```

In [4]:
```
len(df)
```

Out[4]:
```
195
```

In [5]:
```
1 df.dtypes
```

Out[5]:
```
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
```

In [6]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   name              195 non-null    object 
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
 15  NHR               195 non-null    float64
 16  HNR               195 non-null    float64
 17  status            195 non-null    int64  
 18  RPDE              195 non-null    float64
 19  DFA               195 non-null    float64
 20  spread1           195 non-null    float64
 21  spread2           195 non-null    float64
 22  D2                195 non-null    float64
 23  PPE               195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

In [7]:
```
1 df.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
1 df.columns
```

Out[8]:
```
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')
```

In [9]:
```
1 df.isna().sum()
```

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

In [10]:
```
1 df.isnull().sum()
```

Out[10]:
```
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
```

In [11]:
```
1 df.columns.value_counts()
```

Out[11]:
```
name                1
MDVP:Fo(Hz)         1
D2                  1
spread2             1
spread1             1
DFA                 1
RPDE                1
status              1
HNR                 1
NHR                 1
Shimmer:DDA         1
MDVP:APQ            1
Shimmer:APQ5        1
Shimmer:APQ3        1
MDVP:Shimmer(dB)    1
MDVP:Shimmer        1
Jitter:DDP          1
MDVP:PPQ            1
MDVP:RAP            1
MDVP:Jitter(Abs)    1
MDVP:Jitter(%)      1
MDVP:Flo(Hz)        1
MDVP:Fhi(Hz)        1
PPE                 1
dtype: int64
```

In [12]:
```
1 from ydata_profiling import ProfileReport
```

In [21]:
```
1 # ProfileReport(df, title="Parkinson's Disease Profile Report")
```

In [15]:
```
1 df['name'].nunique()
```

Out[15]:
```
195
```

Target Data Column -- status

In [16]:
```
1 df['status']
```

Out[16]:
```
0      1
1      1
2      1
3      1
4      1
      ..
190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```

In [17]:
```
1 df['status'].value_counts()
```

Out[17]:
```
1    147
0     48
Name: status, dtype: int64
```

In [19]:
```
1 sns.histplot(df['status'])
2 plt.show()
```

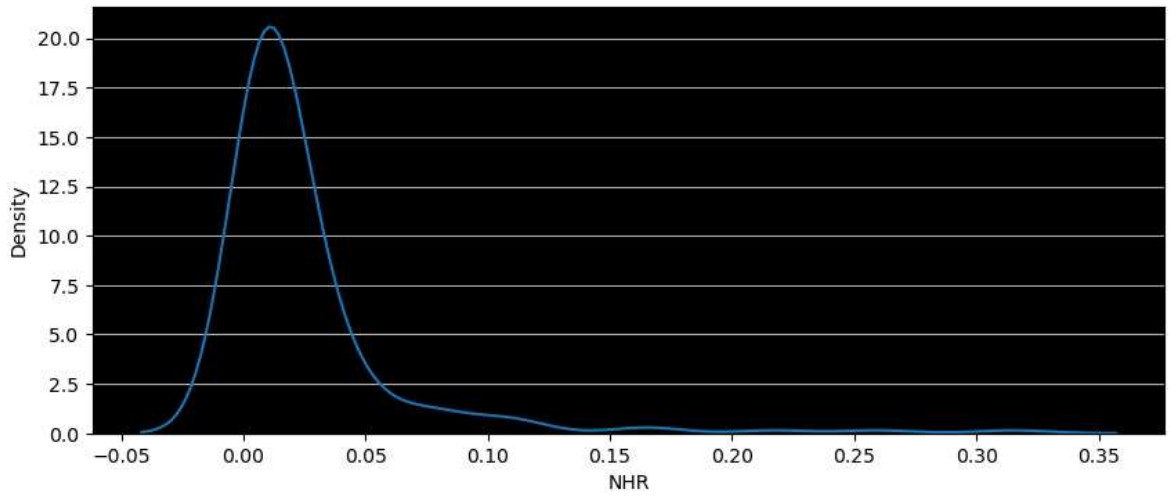
```
C:\Users\HP\AppData\Local\Temp\ipykernel_26056\2761634097.py:2: UserWarning:
Matplotlib is currently using module://matplotlib_inline.backend_inline, whi
ch is a non-GUI backend, so cannot show the figure.
  plt.show()
```

In [20]:
```
1 plt.figure(figsize=(10, 4))
2 sns.histplot(df['status'])
3 plt.xlabel('Status')
4 plt.ylabel('Frequencies')
5 ax=plt.gca()
6 ax.set_facecolor('black')
7 plt.grid()
8 plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_26056\1281180414.py:8: UserWarning:
Matplotlib is currently using module://matplotlib_inline.backend_inline, whi
ch is a non-GUI backend, so cannot show the figure.
  plt.show()
```

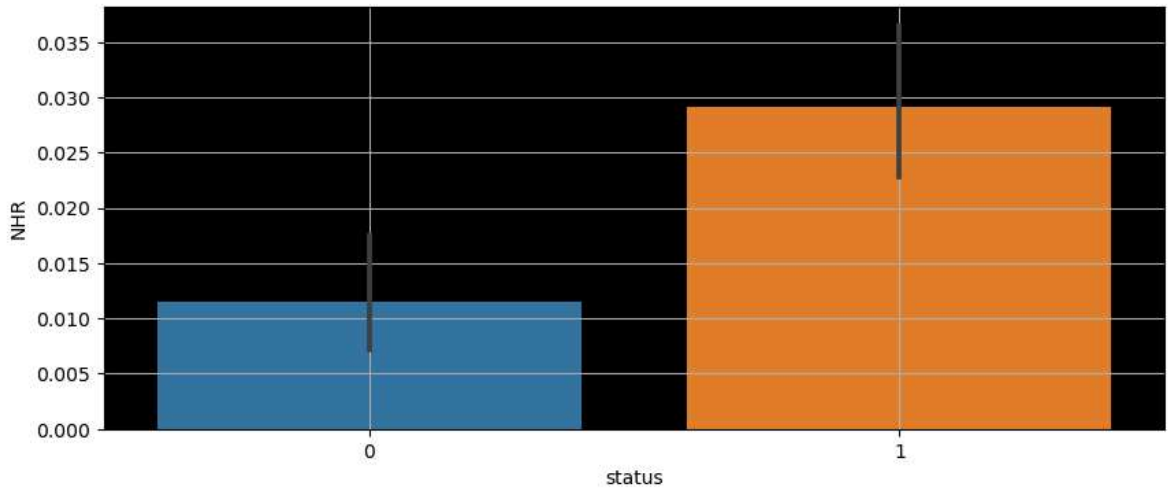
In [17]:

```
plt.figure(figsize=(10, 4))
sns.kdeplot(df['NHR'])
ax=plt.gca()
ax.set_facecolor('black')
plt.grid(axis='y')
plt.show()
```



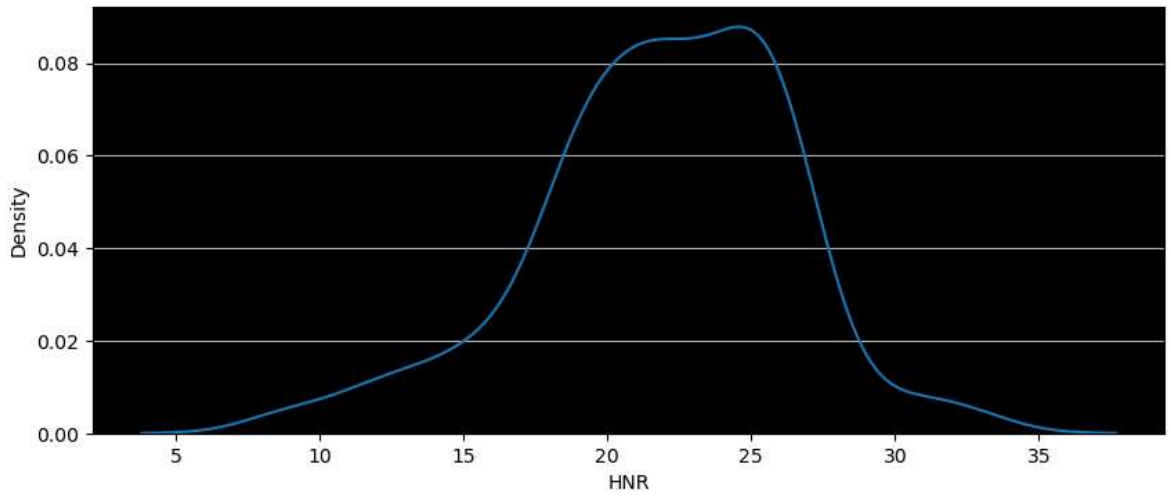
In [18]:

```
plt.figure(figsize=(10, 4))
sns.barplot(x="status",y="NHR",data=df,errorbar=('ci', 95))
plt.grid()
ax=plt.gca()
ax.set_facecolor('black')
plt.show()
```



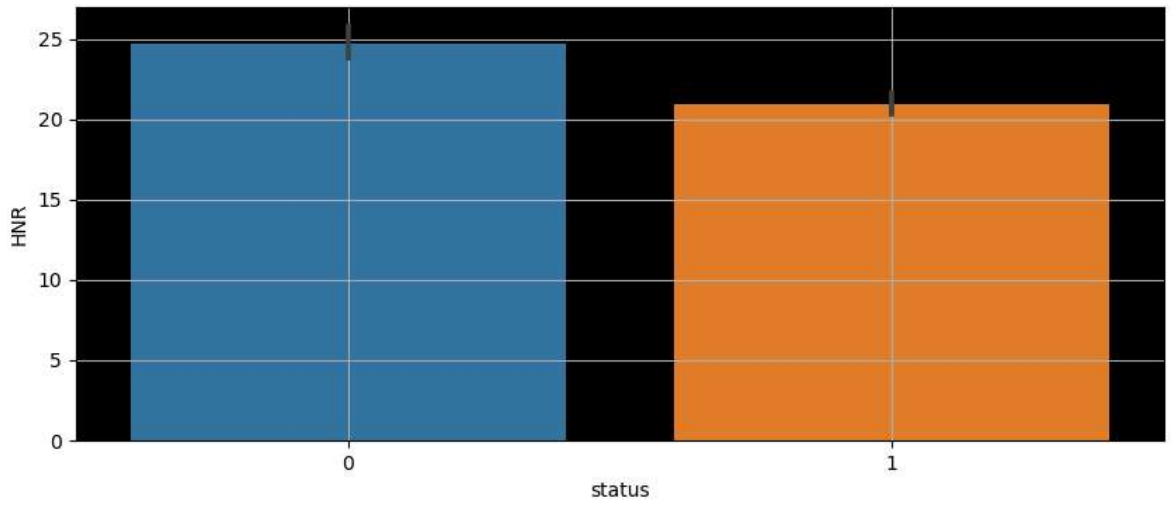
In [19]:

```
plt.figure(figsize=(10, 4))
sns.kdeplot(df['HNR'])
ax=plt.gca()
ax.set_facecolor('black')
plt.grid(axis='y')
plt.show()
```



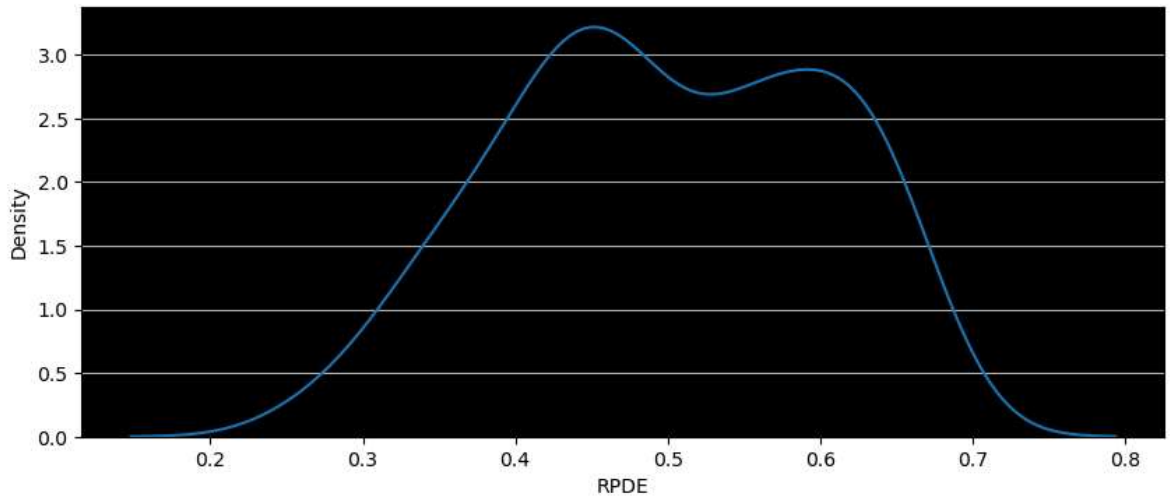
In [20]:

```
plt.figure(figsize=(10,4))
sns.barplot(x="status",y="HNR",data=df)
ax=plt.gca()
ax.set_facecolor('black')
plt.grid()
plt.show()
```



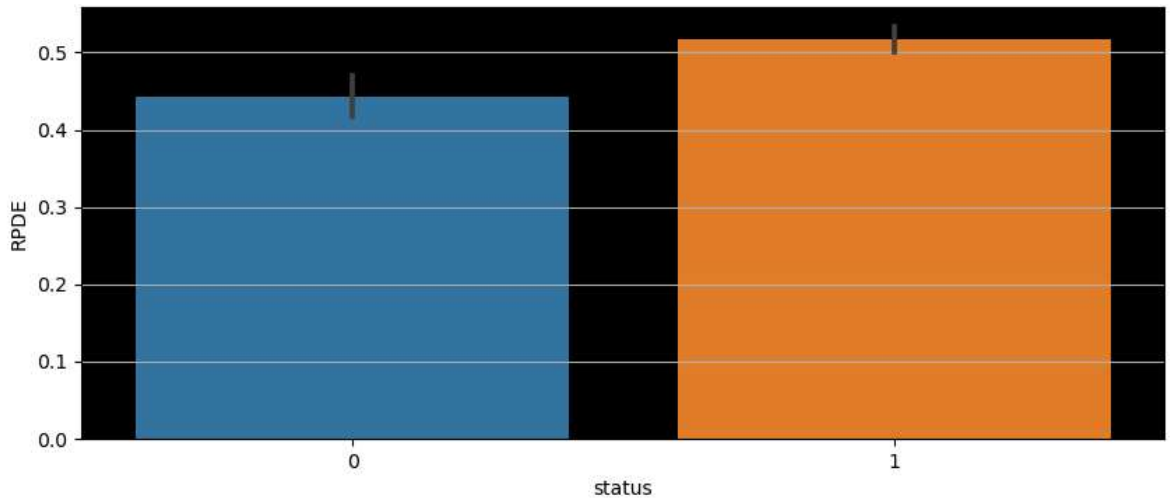
In [21]:

```
plt.figure(figsize=(10,4))
sns.kdeplot(df['RPDE'])
ax=plt.gca()
ax.set_facecolor('black')
plt.grid(axis='y')
plt.show()
```



In [27]:

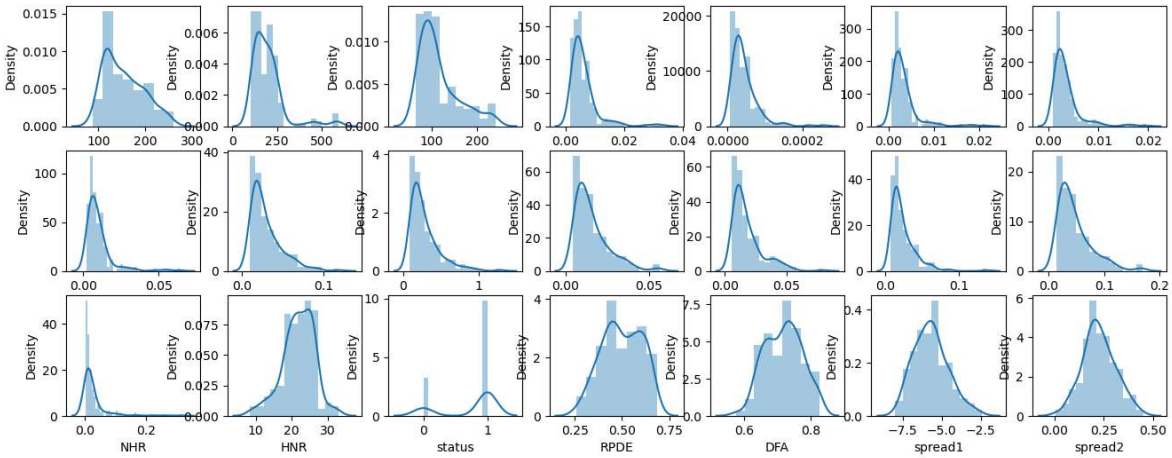
```
plt.figure(figsize=(10,4))
sns.barplot(x="status",y="RPDE",data=df)
ax=plt.gca()
ax.set_facecolor('black')
plt.grid(axis='y')
plt.show()
```



In [28]:

```
import warnings
warnings.filterwarnings('ignore')
rows=3
cols=7
fig, ax=plt.subplots(nrows=rows,ncols=cols,figsize=(16,6))
col=df.columns
index=1
for i in range(rows):
    for j in range(cols):
        sns.distplot(df[col[index]],ax=ax[i][j])
        index=index+1

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



In [29]:
```
corr = df.corr()
corr
```

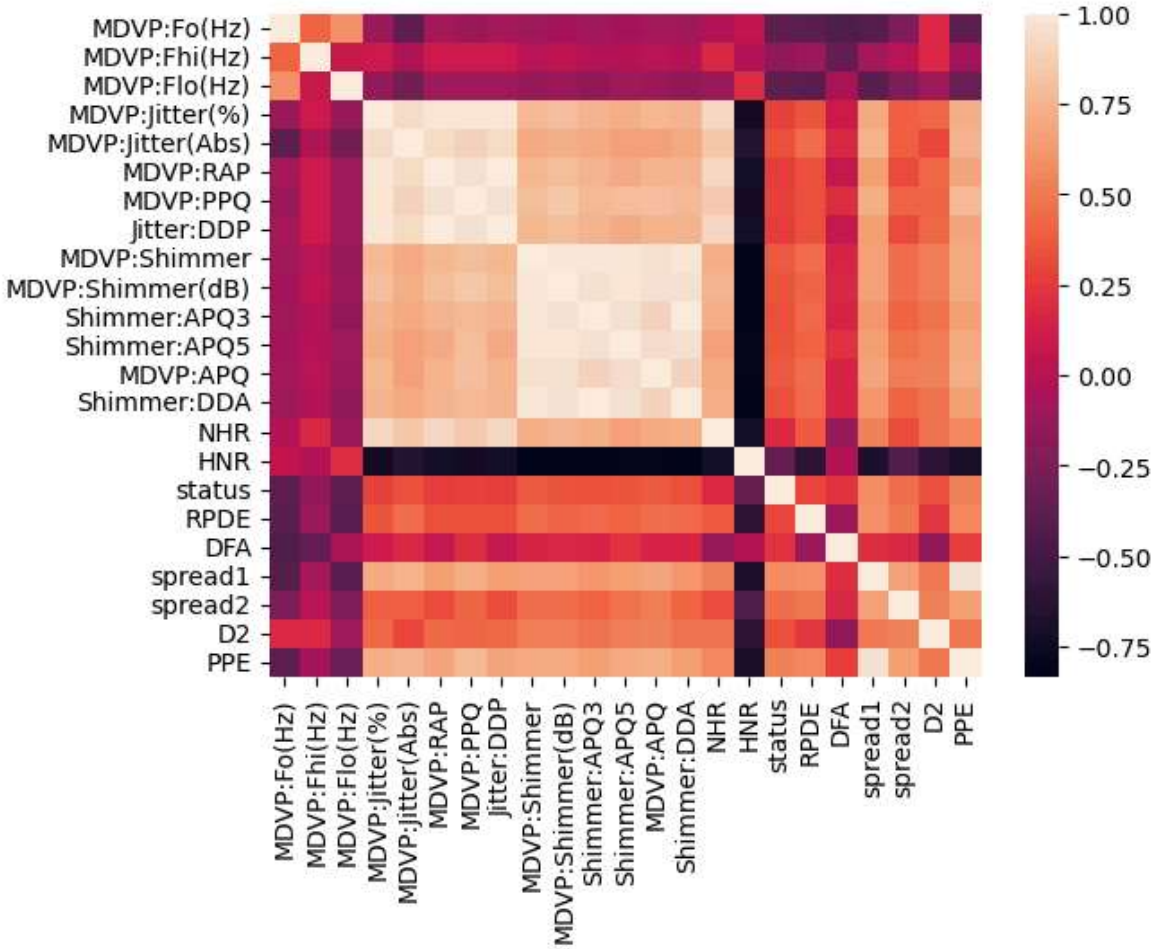
Out[29]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs
MDVP:Fo(Hz)	1.000000	0.400985	0.596546	-0.118003	-0.38202
MDVP:Fhi(Hz)	0.400985	1.000000	0.084951	0.102086	-0.02919
MDVP:Flo(Hz)	0.596546	0.084951	1.000000	-0.139919	-0.27781
MDVP:Jitter(%)	-0.118003	0.102086	-0.139919	1.000000	0.93571
MDVP:Jitter(Abs)	-0.382027	-0.029198	-0.277815	0.935714	1.00000
MDVP:RAP	-0.076194	0.097177	-0.100519	0.990276	0.92291
MDVP:PPQ	-0.112165	0.091126	-0.095828	0.974256	0.89777
Jitter:DDP	-0.076213	0.097150	-0.100488	0.990276	0.92291
MDVP:Shimmer	-0.098374	0.002281	-0.144543	0.769063	0.70332
MDVP:Shimmer(dB)	-0.073742	0.043465	-0.119089	0.804289	0.71660
Shimmer:APQ3	-0.094717	-0.003743	-0.150747	0.746625	0.69715
Shimmer:APQ5	-0.070682	-0.009997	-0.101095	0.725561	0.64896
MDVP:APQ	-0.077774	0.004937	-0.107293	0.758255	0.64879
Shimmer:DDA	-0.094732	-0.003733	-0.150737	0.746635	0.69717
NHR	-0.021981	0.163766	-0.108670	0.906959	0.83497
HNR	0.059144	-0.024893	0.210851	-0.728165	-0.65681
status	-0.383535	-0.166136	-0.380200	0.278220	0.33865
RPDE	-0.383894	-0.112404	-0.400143	0.360673	0.44183
DFA	-0.446013	-0.343097	-0.050406	0.098572	0.17503
spread1	-0.413738	-0.076658	-0.394857	0.693577	0.73577
spread2	-0.249450	-0.002954	-0.243829	0.385123	0.38854
D2	0.177980	0.176323	-0.100629	0.433434	0.31069
PPE	-0.372356	-0.069543	-0.340071	0.721543	0.74816

23 rows × 23 columns

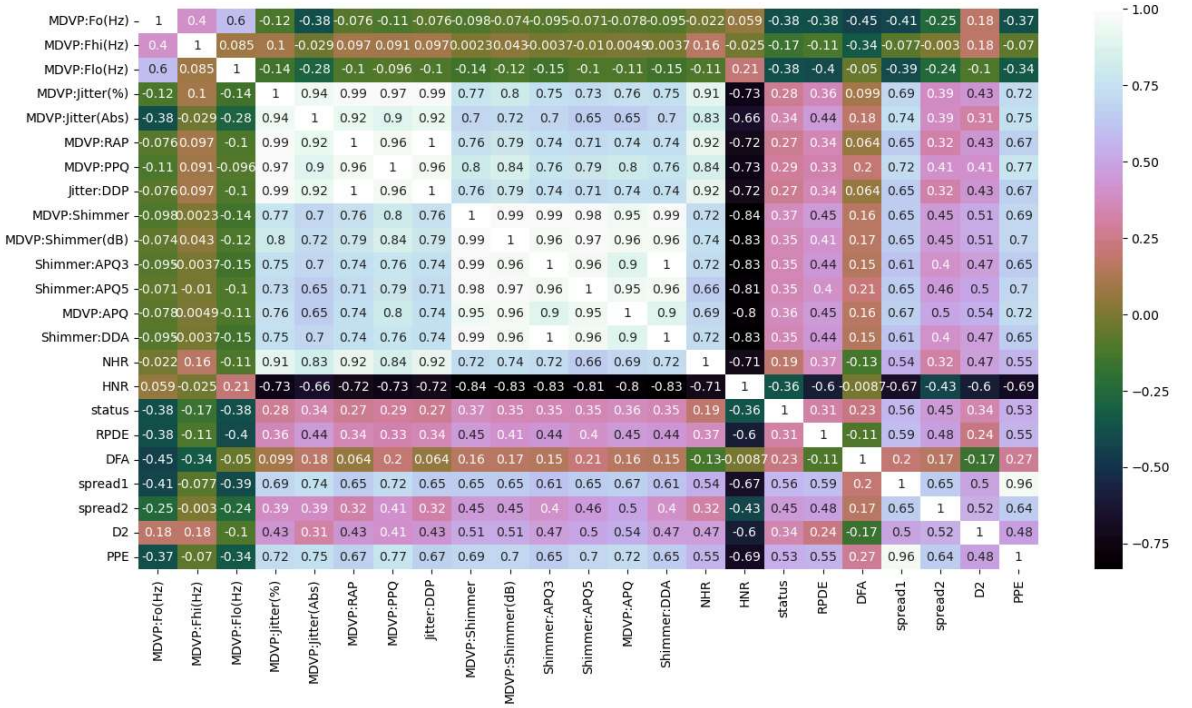
In [30]:

```
1 sns.heatmap(corr)
2 plt.show()
```



Type *Markdown* and LaTeX: α^2

In [31]:
```
plt.figure(figsize=(16,8))
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, cma
plt.show()
```



Machine Learnig Algorithm

In [32]:
```
df.head()
```

Out[32]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.0000
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.0000
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.0000
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.0000
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.0001

5 rows × 24 columns

The name column from the data set does not play a significant role for the machine learning analysis. Based on the aforementioned information set for the Parkinson's Diesase test, we must determine the test for both independent and dependent variables. Here, the status will play a dependent variable and the other columns will play for the independent variables to check how the fitness for the played variables for independent lyes for testing the status.

In [33]:
```
df.drop(columns=['name'], inplace=True)
```

Creating The independent and dependent columns for testing for machine learning

In [35]:
```
X=df.drop(labels=['status'],axis=1)
X.head()
#Below all are the columns for the independent testing
```

Out[35]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MD
0	119.992	157.302	74.997	0.00784	0.00007	0.00370	
1	122.400	148.650	113.819	0.00968	0.00008	0.00465	
2	116.682	131.111	111.555	0.01050	0.00009	0.00544	
3	116.676	137.871	111.366	0.00997	0.00009	0.00502	
4	116.014	141.781	110.655	0.01284	0.00011	0.00655	

5 rows × 22 columns

In [36]:
```
Y=df['status']
Y.head()
#Below all are the columns for the dependent for analysing for testing
```

Out[36]:
```
0    1
1    1
2    1
3    1
4    1
Name: status, dtype: int64
```

In [37]:
```
Y.value_counts()
```

Out[37]:
```
1    147
0     48
Name: status, dtype: int64
```

In [38]:
```
print (X.shape,Y.shape)
```

```
(195, 22) (195,)
```

Training and Testing

In [39]:
```
1 X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,ran
```

In [40]:
```
1 print(X_train.shape)
2 print(X_test.shape)
```

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

In [41]:
```
1 print(Y_train.shape)
2 print(Y_test.shape)
```

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

Checking the Data with the Logistic Regression

In [42]:
```
1 log_reg = LogisticRegression()
2 log_reg
```

Out[42]:
```
LogisticRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [43]:
```
1 log_reg.fit(X_train, Y_train)
```

Out[43]:
```
LogisticRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Prediction on train

In [44]:
```
1 Y_train[:10]
```

Out[44]:
```
147    1
86     1
179    1
69     1
125    1
42     0
77     1
62     0
153    1
59     1
Name: status, dtype: int64
```

In [45]:
```
1 train_preds = log_reg.predict(X_train)
2 train_preds
```

Out[45]:
```
array([1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1], dtype=int64)
```

In [46]:
```
1 print("Model accuracy on train is: ", accuracy_score(Y_train,train_preds))
```

```
Model accuracy on train is:  0.8717948717948718
```

In [47]:
```
1 test_preds = log_reg.predict(X_test)
2 test_preds
```

Out[47]:
```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1], dtype=int64)
```

In [48]:
```
1 Y_test
```

Out[48]:
```
96     1
5      1
116    1
35     0
178    1
185    0
54     1
134    1
90     1
187    0
139    1
142    1
175    0
26     1
89     1
140    1
155    1
23     1
132    1
37     1
151    1
28     1
85     1
93     1
172    0
75     1
18     1
105    1
121    1
130    1
33     0
46     0
166    0
163    1
11     1
164    1
81     1
111    1
67     1
Name: status, dtype: int64
```

In [49]:
```
1 print("Model accuracy on test is: ", accuracy_score(Y_test, test_preds))
2 print('+'*50)
```

```
Model accuracy on test is:  0.8461538461538461
++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [50]:
```
1 print("confusion_matrix train is:\n ", confusion_matrix(Y_train, train_pr
```

```
confusion_matrix train is:
  [[ 24  16]
 [  4 112]]
```

In [51]:
```
print("confusion_matrix test is:\n ", confusion_matrix(Y_test, test_preds
```

```
confusion_matrix test is:
  [[ 5  3]
 [ 3 28]]
```

In [52]:
```
print('\nClassification Report Train is ')
print(classification_report (Y_train, train_preds))
```

```
Classification Report Train is
              precision    recall  f1-score   support

           0       0.86      0.60      0.71        40
           1       0.88      0.97      0.92       116

    accuracy                           0.87       156
   macro avg       0.87      0.78      0.81       156
weighted avg       0.87      0.87      0.86       156
```

In [53]:
```
print('\nClassification Report Train is ')
print(classification_report (Y_test, test_preds))
```

```
Classification Report Train is
              precision    recall  f1-score   support

           0       0.62      0.62      0.62         8
           1       0.90      0.90      0.90        31

    accuracy                           0.85        39
   macro avg       0.76      0.76      0.76        39
weighted avg       0.85      0.85      0.85        39
```

Checking for the Random Forest Model

In [54]:
```
Rf = RandomForestClassifier()
Rf
```

Out[54]:
```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [55]:
```
1 Rf.fit(X_train,Y_train)
```

Out[55]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [56]:
```
1 train_predsRf = Rf.predict(X_train)
2 train_predsRf
```

Out[56]:
```
array([1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1], dtype=int64)
```

In [57]:
```
1 print("Model accuracy on train is:",accuracy_score(Y_train, train_predsRf
```

```
Model accuracy on train is: 1.0
```

In [58]:
```
1 test_predsRf  = Rf.predict(X_test)
2 test_predsRf
```

Out[58]:
```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1], dtype=int64)
```

In [59]:
```
1 print("Model accuracy on train is:",accuracy_score(Y_test,test_predsRf))
```

```
Model accuracy on train is: 0.8974358974358975
```

In [60]:
```
1 print("confusion_matrix train is:\n ", confusion_matrix(Y_train, train_pr
```

```
confusion_matrix train is:
  [[ 40   0]
 [  0 116]]
```

In [61]:
```
1 print("confusion_matrix train is:\n ", confusion_matrix(Y_test, test_pred
```

```
confusion_matrix train is:
  [[ 6  2]
 [ 2 29]]
```

In [62]:
```
print('\nClassification Report Train is ')
print(classification_report (Y_train, train_predsRf))
```

```
Classification Report Train is
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00       116

    accuracy                           1.00       156
   macro avg       1.00      1.00      1.00       156
weighted avg       1.00      1.00      1.00       156
```

In [63]:
```
print('\nClassification Report Train is ')
print(classification_report (Y_test, test_predsRf))
```

```
Classification Report Train is
              precision    recall  f1-score   support

           0       0.75      0.75      0.75         8
           1       0.94      0.94      0.94        31

    accuracy                           0.90        39
   macro avg       0.84      0.84      0.84        39
weighted avg       0.90      0.90      0.90        39
```

In [64]:
```
print((Y_test !=test_predsRf).sum(),'/',((Y_test == test_predsRf).sum()+(
```

```
4 / 39
```

In [65]:
```
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test,test_predsRf))
```

```
KappaScore is:  0.685483870967742
```

Final Result we got from Randomforest Classifier

Below is the Data Frame for expected and the test Results

In [66]:
```
1 ddf=pd.DataFrame(data=[test_predsRf,Y_test])
2 display (ddf)
```

	0	1	2	3	4	5	6	7	8	9	...	29	30	31	32	33	34	35	36	37	38
0	1	1	1	0	1	1	1	1	1	1	...	1	0	0	0	1	1	1	1	0	1
1	1	1	1	0	1	0	1	1	1	0	...	1	0	0	0	1	1	1	1	1	1

2 rows × 39 columns

In [67]:

```
1 ddf.T
```

Out[67]:

	0	1
0	1	1
1	1	1
2	1	1
3	0	0
4	1	1
5	1	0
6	1	1
7	1	1
8	1	1
9	1	0
10	1	1
11	1	1
12	0	0
13	1	1
14	1	1
15	1	1
16	1	1
17	1	1
18	1	1
19	1	1
20	1	1
21	0	1
22	1	1
23	1	1
24	0	0
25	1	1
26	1	1
27	1	1
28	1	1
29	1	1
30	0	0
31	0	0
32	0	0
33	1	1
34	1	1
35	1	1

	0	1
36	1	1
37	0	1
38	1	1

Analysing for Decision Tree classifier

In [68]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [69]:
```
DT = DecisionTreeClassifier()
DT
```

Out[69]:
```
DecisionTreeClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [70]:
```
DT.fit(X_train,Y_train)
```

Out[70]:
```
DecisionTreeClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Training the Models`

In [71]:
```
train_predsDT=DT.predict(X_train)
train_predsDT
```

Out[71]:
```
array([1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1], dtype=int64)
```

In [72]:
```
print("Model accuracy on train is:",accuracy_score(Y_train,train_predsDT)
print('+'*50)
```

```
Model accuracy on train is: 1.0
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Testing the Models

In [73]:
```
test_predsDT = DT.predict(X_test)
test_predsDT
```

Out[73]:
```
array([1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1], dtype=int64)
```

In [74]:
```
print("Model accuracy on train is:",accuracy_score(Y_test,test_predsDT))
print('+'*50)
```

```
Model accuracy on train is: 0.9230769230769231
++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [75]:
```
print("confusion_matrix train is:\n ", confusion_matrix(Y_train, train_pr
```

```
confusion_matrix train is:
  [[ 40   0]
 [  0 116]]
```

In [76]:
```
print("confusion_matrix test is: \n", confusion_matrix(Y_test, test_preds
```

```
confusion_matrix test is:
 [[ 8  0]
 [ 3 28]]
```

In [77]:
```
print('Wrong predictions out of total')
print('-'*50)
print('\nClassification Report Train is ')
```

```
Wrong predictions out of total
--------------------------------------------------

Classification Report Train is
```

In [78]:
```
print(classification_report (Y_train, train_predsDT))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00       116

    accuracy                           1.00       156
   macro avg       1.00      1.00      1.00       156
weighted avg       1.00      1.00      1.00       156
```

In [79]:
```
print('\nClassification Report Test is ')
print(classification_report (Y_test, test_predsDT))
```

```
Classification Report Test is
              precision    recall  f1-score   support

           0       0.73      1.00      0.84         8
           1       1.00      0.90      0.95        31

    accuracy                           0.92        39
   macro avg       0.86      0.95      0.90        39
weighted avg       0.94      0.92      0.93        39
```

In [80]:
```
print((Y_test !=test_predsDT).sum(),'/',((Y_test == test_predsDT).sum()+(
print('+'*50)
```

```
3 / 39
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Kappa Score

In [81]:
```
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test,test_predsDT))
```

```
KappaScore is:  0.7929203539823009
```

Analysing using Naive Bayce algorithm

In [82]:
```
from sklearn.naive_bayes import GaussianNB
```

In [83]:
```
NB = GaussianNB()
NB
```

Out[83]:
```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Training the Models`

In [84]:
```
1 NB.fit(X_train, Y_train)
```

Out[84]:
```
GaussianNB()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [85]:
```
1 train_predsNB = NB.predict(X_train)
2 train_predsNB
```

Out[85]:
```
array([1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1], dtype=int64)
```

In [86]:
```
1 print("Model accuracy on train is:",accuracy_score(Y_train,train_predsNB)
2 print('+'*50)
```

```
Model accuracy on train is: 0.7307692307692307
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Testing the Models

In [87]:
```
1 test_predsNB = NB.predict(X_test)
2 test_predsNB
```

Out[87]:
```
array([1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1], dtype=int64)
```

In [88]:
```
1 print("Model accuracy on train is:",accuracy_score(Y_test,test_predsNB))
2 print('+'*50)
```

```
Model accuracy on train is: 0.6923076923076923
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Confusion matrix

In [89]:
```
print("confusion_matrix train is: \n", confusion_matrix(Y_train, train_pr
```

```
confusion_matrix train is:
 [[38  2]
 [40 76]]
```

In [90]:
```
print("confusion_matrix test is:\n ", confusion_matrix(Y_test, test_preds
```

```
confusion_matrix test is:
  [[ 8  0]
 [12 19]]
```

In [91]:
```
print('Wrong predictions out of total')
print('+'*50)
```

```
Wrong predictions out of total
++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [92]:
```
print('\nClassification Report Train is \n')
print(classification_report (Y_train, train_predsNB))
```

```
Classification Report Train is

              precision    recall  f1-score   support

           0       0.49      0.95      0.64        40
           1       0.97      0.66      0.78       116

    accuracy                           0.73       156
   macro avg       0.73      0.80      0.71       156
weighted avg       0.85      0.73      0.75       156
```

In [93]:
```
print('\nClassification Report Test is ')
print(classification_report (Y_test, test_predsNB))
```

```
Classification Report Test is
              precision    recall  f1-score   support

           0       0.40      1.00      0.57         8
           1       1.00      0.61      0.76        31

    accuracy                           0.69        39
   macro avg       0.70      0.81      0.67        39
weighted avg       0.88      0.69      0.72        39
```

In [94]:
```
print((Y_test !=test_predsNB).sum(),'/',((Y_test == test_predsNB).sum()+(
print('+'*10)
```

```
12 / 39
++++++++++
```

Kappa Score

In [95]:
```
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test,test_predsNB))
```

```
KappaScore is:  0.3937823834196892
```

Analysing the Models using K Neighbours Classifier

In [96]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [97]:
```
KNN = KNeighborsClassifier()
KNN
```

Out[97]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Training the Data Models

In [98]:
```
KNN.fit(X_train,Y_train)
```

Out[98]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Predicting Train Models

In [99]:
```
train_predsKNN = KNN.predict(X_train)
train_predsKNN
```

Out[99]:
```
array([1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       1, 1], dtype=int64)
```

In [100]:
```
print("Model accuracy on train is: ", accuracy_score(Y_train, train_preds
print("+"*50)
```

```
Model accuracy on train is:  0.9102564102564102
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Predicting Test Models

In [101]:
```
test_predsKNN = KNN.predict(X_test)
test_predsKNN
```

Out[101]:
```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1], dtype=int64)
```

In [102]:
```
1 Y_test
```

Out[102]:
```
96     1
5      1
116    1
35     0
178    1
185    0
54     1
134    1
90     1
187    0
139    1
142    1
175    0
26     1
89     1
140    1
155    1
23     1
132    1
37     1
151    1
28     1
85     1
93     1
172    0
75     1
18     1
105    1
121    1
130    1
33     0
46     0
166    0
163    1
11     1
164    1
81     1
111    1
67     1
Name: status, dtype: int64
```

In [103]:
```
1 print("Model accuracy on train is: ", accuracy_score(Y_test,test_preds))
2 print("+"*50)
```

```
Model accuracy on train is:  0.8461538461538461
++++++++++++++++++++++++++++++++++++++++++++++++++
```

confusion Matrix

In [104]:
```
print("confusion_matrix train is:\n ", confusion_matrix(Y_train, train_pr
```

```
confusion_matrix train is:
  [[ 30  10]
 [  4 112]]
```

In [105]:
```
print("confusion_matrix test is:\n ", confusion_matrix(Y_test, test_preds
```

```
confusion_matrix test is:
  [[ 4  4]
 [ 2 29]]
```

In [106]:
```
print('Wrong predictions out of total')
print('+'*50)
print('\nClassification Report Train is ')
```

```
Wrong predictions out of total
++++++++++++++++++++++++++++++++++++++++++++++++++

Classification Report Train is
```

In [107]:
```
print(classification_report (Y_train, train_predsKNN))
```

```
              precision    recall  f1-score   support

           0       0.88      0.75      0.81        40
           1       0.92      0.97      0.94       116

    accuracy                           0.91       156
   macro avg       0.90      0.86      0.88       156
weighted avg       0.91      0.91      0.91       156
```

In [108]:
```
print('\nClassification Report Test is \n')
print(classification_report (Y_test, test_predsKNN))
```

```
Classification Report Test is

              precision    recall  f1-score   support

           0       0.67      0.50      0.57         8
           1       0.88      0.94      0.91        31

    accuracy                           0.85        39
   macro avg       0.77      0.72      0.74        39
weighted avg       0.84      0.85      0.84        39
```

In [109]:
```
print((Y_test !=test_predsKNN).sum(),'/',((Y_test == test_predsKNN).sum()
print("+"*20)
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test,test_predsKNN))
```

```
6 / 39
++++++++++++++++++++
KappaScore is:  0.48
```

Analysing from Support Vector Machine Classifier

In [110]:
```
from sklearn.svm import SVC
```

In [111]:
```
SVM = SVC(kernel='linear')
SVM
```

Out[111]:
```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Training Data Set models

In [112]:
```
SVM.fit(X_train, Y_train)
```

Out[112]:
```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [113]:
```
train_predsSVM = SVM.predict(X_train)
train_predsSVM
```

Out[113]:
```
array([1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1], dtype=int64)
```

In [114]:
```
print("Model accuracy on train is: ", accuracy_score(Y_train, train_preds
print("+"*50)
```

```
Model accuracy on train is:  0.8782051282051282
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Testing the models

In [115]:
```
test_predsSMV = SVM.predict(X_test)
test_predsSMV
```

Out[115]:
```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [116]:
```
print("Model accuracy on train is: ", accuracy_score(Y_test,test_predsSMV
print("+"*50)
```

```
Model accuracy on train is:  0.8974358974358975
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Confusion matrix

In [117]:
```
print("Confusion matrix for the Training Data models: \n" ,confusion_matr
```

```
Confusion matrix for the Training Data models:
 [[ 23  17]
 [  2 114]]
```

In [121]:
```
print("Confusion matrix for the Testing Data models: \n" ,confusion_matri
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[121], line 1
----> 1 print("Confusion matrix for the Testing Data models: \n" ,confusion_
matrix(Y_test,test_predsSMv))

NameError: name 'test_predsSMv' is not defined
```

In [119]:
```
print('Wrong predictions out of total')
print('-'*30)

print("recall", metrics.recall_score(Y_test, test_predsSMV))
print('-'*30)

```

```
Wrong predictions out of total
------------------------------
recall 0.967741935483871
------------------------------
```

In [120]:
```
print('\nClassification Report Train is \n')
print(classification_report (Y_train, train_predsSVM))
```

```
Classification Report Train is

              precision    recall  f1-score   support

           0       0.92      0.57      0.71        40
           1       0.87      0.98      0.92       116

    accuracy                           0.88       156
   macro avg       0.90      0.78      0.82       156
weighted avg       0.88      0.88      0.87       156
```

In [ ]:
```
print('\nClassification Report Test is \n')
print(classification_report (Y_test, test_predsSMV))
```

In [ ]:
```
print((Y_test !=test_predsSMV).sum(),'/',((Y_test == test_predsSMV).sum()
```

Kappa Score

In [122]:
```
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test,test_predsSMV))
```

```
KappaScore is:  0.6533333333333333
```