In [2]:

```
import os, sys
import numpy as np
import pandas as pd
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
import joblib
```

In [6]:

```
parkinson_df= pd.read_csv('parkinsons.csv')
pd.set_option("display.max_columns", None)

parkinson_df.head(5)
```

Out[6]:

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

In [7]:

```
parkinson_df.shape
```

Out[7]: (195, 24)

In [9]:

```
parkinson_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 [10]:

```
parkinson_df.describe()
```

Out[10]:

Out[10]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVF
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	

In [11]:

```
parkinson_df.corr()
```

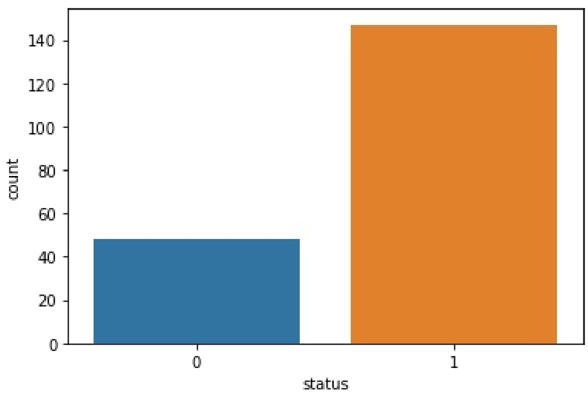
Out[11]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shi
MDVP:Fo(Hz)	1.000000	0.400985	0.596546	-0.118003	-0.382027	-0.076194	-0.112165	-0.076213	-0.0
MDVP:Fhi(Hz)	0.400985	1.000000	0.084951	0.102086	-0.029198	0.097177	0.091126	0.097150	0.0
MDVP:Flo(Hz)	0.596546	0.084951	1.000000	-0.139919	-0.277815	-0.100519	-0.095828	-0.100488	-0.1
MDVP:Jitter(%)	-0.118003	0.102086	-0.139919	1.000000	0.935714	0.990276	0.974256	0.990276	0.7
MDVP:Jitter(Abs)	-0.382027	-0.029198	-0.277815	0.935714	1.000000	0.922911	0.897778	0.922913	0.7
MDVP:RAP	-0.076194	0.097177	-0.100519	0.990276	0.922911	1.000000	0.957317	1.000000	0.7
MDVP:PPQ	-0.112165	0.091126	-0.095828	0.974256	0.897778	0.957317	1.000000	0.957319	0.7
Jitter:DDP	-0.076213	0.097150	-0.100488	0.990276	0.922913	1.000000	0.957319	1.000000	0.7
MDVP:Shimmer	-0.098374	0.002281	-0.144543	0.769063	0.703322	0.759581	0.797826	0.759555	1.0
MDVP:Shimmer(dB)	-0.073742	0.043465	-0.119089	0.804289	0.716601	0.790652	0.839239	0.790621	0.9
Shimmer:APQ3	-0.094717	-0.003743	-0.150747	0.746625	0.697153	0.744912	0.763580	0.744894	0.9
Shimmer:APQ5	-0.070682	-0.009997	-0.101095	0.725561	0.648961	0.709927	0.786780	0.709907	0.9
MDVP:APQ	-0.077774	0.004937	-0.107293	0.758255	0.648793	0.737455	0.804139	0.737439	0.9
Shimmer:DDA	-0.094732	-0.003733	-0.150737	0.746635	0.697170	0.744919	0.763592	0.744901	0.9
NHR	-0.021981	0.163766	-0.108670	0.906959	0.834972	0.919521	0.844604	0.919548	0.7
HNR	0.059144	-0.024893	0.210851	-0.728165	-0.656810	-0.721543	-0.731510	-0.721494	-0.8
status	-0.383535	-0.166136	-0.380200	0.278220	0.338653	0.266668	0.288698	0.266646	0.3
RPDE	-0.383894	-0.112404	-0.400143	0.360673	0.441839	0.342140	0.333274	0.342079	0.4
DFA	-0.446013	-0.343097	-0.050406	0.098572	0.175036	0.064083	0.196301	0.064026	0.1
spread1	-0.413738	-0.076658	-0.394857	0.693577	0.735779	0.648328	0.716489	0.648328	0.6
spread2	-0.249450	-0.002954	-0.243829	0.385123	0.388543	0.324407	0.407605	0.324377	0.4
D2	0.177980	0.176323	-0.100629	0.433434	0.310694	0.426605	0.412524	0.426556	0.5
PPE	-0.372356	-0.069543	-0.340071	0.721543	0.748162	0.670999	0.769647	0.671005	0.6

In [13]:

```
sns.countplot(parkinson_df['status'])
```

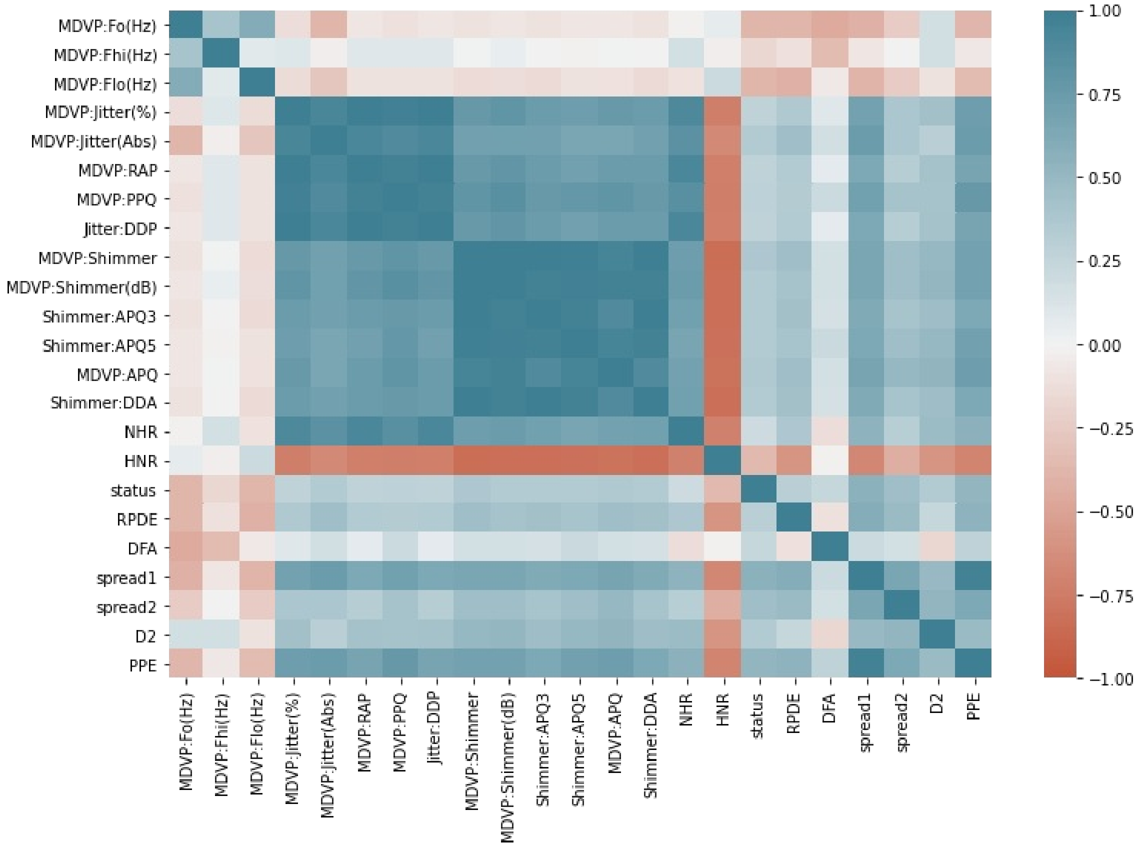
Out[13]: <AxesSubplot:xlabel='status', ylabel='count'>



In [16]:

```
fig, ax = plt.subplots(figsize=(12, 8))
```

```python
fig, ax = plt.subplots(figsize=(12, 8))
corr = parkinson_df.corr()
ax = sns.heatmap(corr, vmin=-1, vmax=1, center=0, cmap=sns.diverging_palette(20, 220, n=200))
```



In [17]:

```python
#Rearrange the columns
parkinson_df = parkinson_df[["name", "MDVP:Fo(Hz)", "MDVP:Fhi(Hz)", "MDVP:Flo(Hz)", "MDVP:Jitter(%)", "MDVP:Jitte


#Create a copy of the original dataset
df2= parkinson_df.copy()

#Assign numeric values to the binary and categorical columns
number= LabelEncoder()
df2['name']= number.fit_transform(df2['name'])

df2.head(5)
```

Out[17]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MD
0	0	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	
1	1	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	
2	2	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	
3	3	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	
4	4	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	

In [21]:

```python
X= df2.iloc[:,0:11] #all features
Y= df2.iloc[:,-1] #target (status of Parkinson)

best_features= SelectKBest(score_func=chi2, k=3) #function that select the top 3 features.
fit= best_features.fit(X,Y)

#Creating dataframes for the features and the score of each feature.
Parkinson_scores= pd.DataFrame(fit.scores_)
Parkinson_columns= pd.DataFrame(X.columns)
```

In [22]:

```python
#Create a dataframe that combines all the features and their corresponding scores.
features_scores= pd.concat([Parkinson_scores, Parkinson_scores], axis=1)
features_scores.columns= ['Features', 'Score']
features_scores.sort_values(by = 'Score')
```

Out[22]:

	Features	Score
5	0.000614	0.000614

7	0.035713	0.035713
6	0.036749	0.036749
4	0.056742	0.056742
8	0.110222	0.110222
9	0.313475	0.313475
10	3.210348	3.210348
0	178.712392	178.712392
2	227.402656	227.402656
1	316.985398	316.985398
3	456.626628	456.626628

From the correlation heatmap and feature selection step we conclude that the 3 most affecting features on the target out put are: 1- MDVP:Flo(Hz) 2- MDVP:Fo(Hz) 3- MDVP:Fhi(Hz)

In [23]:

```
x= parkinson_df[["MDVP:Flo(Hz)", "MDVP:Fo(Hz)", "MDVP:Fhi(Hz)"]]
y= parkinson_df[["status"]]
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2, random_state=7)

model=XGBClassifier()
model.fit(x_train,y_train)
```

Out[23]:

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

In [24]:

```
y_pred=model.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
```

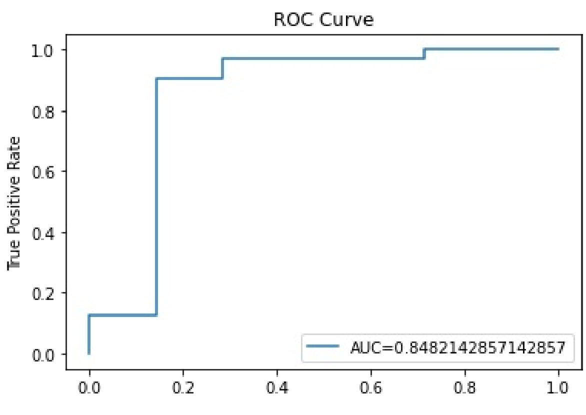
```
87.17948717948718
```

In [25]:

```
#define metrics
y_pred_proba= model.predict_proba(x_test) [::,1]

#Calculate true positive and false positive rates
false_positive_rate, true_positive_rate, _ = metrics.roc_curve(y_test, y_pred_proba)

#Calculate the area under curve to see the model performance
auc= metrics.roc_auc_score(y_test, y_pred_proba)

#Create ROC curve
plt.plot(false_positive_rate, true_positive_rate,label="AUC="+str(auc))
plt.title('ROC Curve')
plt.ylabel('True Positive Rate')
plt.xlabel('false Positive Rate')
plt.legend(loc=4)
```

Out[25]:

```
<matplotlib.legend.Legend at 0x7fea6b167790>
```



false Positive Rate

The area under the curve (AUC) is 0.84, which is very close to one, meaning that the model did a good job.

In [26]:

```
# Save the trained model to a file to be used in future predictions
joblib.dump(model, 'XG.pkl')
```

Out[26]: ['XG.pkl']

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js