```
In [71]: import numpy as np
         import pandas as pd
         import os, sys
         import matplotlib.pyplot as plt
         %matplotlib inline
         import seaborn as sns
         from sklearn.preprocessing import MinMaxScaler
         from xgboost import XGBClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score
```

```
In [72]: df=pd.read_csv("parkinsons.data")
```

```
In [73]: df.head()
```

Out[73]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470

5 rows × 24 columns

```
In [74]: # now remove the status from data set and add it in to the last
         df1=df.pop('status')
         df['status']=df1
```

In [75]:
```
# by using describe we caluclating the count,mean,std,min,25%,50%,75%,max...:
df.describe().transpose()
```

Out[75]:

	count	mean	std	min	25%	50%	75%	max
MDVP:Fo(Hz)	195.0	154.228641	41.390065	88.333000	117.572000	148.790000	182.769000	260.105000
MDVP:Fhi(Hz)	195.0	197.104918	91.491548	102.145000	134.862500	175.829000	224.205500	592.030000
MDVP:Flo(Hz)	195.0	116.324631	43.521413	65.476000	84.291000	104.315000	140.018500	239.170000
MDVP:Jitter(%)	195.0	0.006220	0.004848	0.001680	0.003460	0.004940	0.007365	0.033160
MDVP:Jitter(Abs)	195.0	0.000044	0.000035	0.000007	0.000020	0.000030	0.000060	0.000260
MDVP:RAP	195.0	0.003306	0.002968	0.000680	0.001660	0.002500	0.003835	0.021440
MDVP:PPQ	195.0	0.003446	0.002759	0.000920	0.001860	0.002690	0.003955	0.019580
Jitter:DDP	195.0	0.009920	0.008903	0.002040	0.004985	0.007490	0.011505	0.064330
MDVP:Shimmer	195.0	0.029709	0.018857	0.009540	0.016505	0.022970	0.037885	0.119080
MDVP:Shimmer(dB)	195.0	0.282251	0.194877	0.085000	0.148500	0.221000	0.350000	1.302000
Shimmer:APQ3	195.0	0.015664	0.010153	0.004550	0.008245	0.012790	0.020265	0.056470
Shimmer:APQ5	195.0	0.017878	0.012024	0.005700	0.009580	0.013470	0.022380	0.079400
MDVP:APQ	195.0	0.024081	0.016947	0.007190	0.013080	0.018260	0.029400	0.137780
Shimmer:DDA	195.0	0.046993	0.030459	0.013640	0.024735	0.038360	0.060795	0.169420
NHR	195.0	0.024847	0.040418	0.000650	0.005925	0.011660	0.025640	0.314820
HNR	195.0	21.885974	4.425764	8.441000	19.198000	22.085000	25.075500	33.047000
RPDE	195.0	0.498536	0.103942	0.256570	0.421306	0.495954	0.587562	0.685151
DFA	195.0	0.718099	0.055336	0.574282	0.674758	0.722254	0.761881	0.825288
spread1	195.0	-5.684397	1.090208	-7.964984	-6.450096	-5.720868	-5.046192	-2.434031
spread2	195.0	0.226510	0.083406	0.006274	0.174351	0.218885	0.279234	0.450493
D2	195.0	2.381826	0.382799	1.423287	2.099125	2.361532	2.636456	3.671155
PPE	195.0	0.206552	0.090119	0.044539	0.137451	0.194052	0.252980	0.527367
status	195.0	0.753846	0.431878	0.000000	1.000000	1.000000	1.000000	1.000000

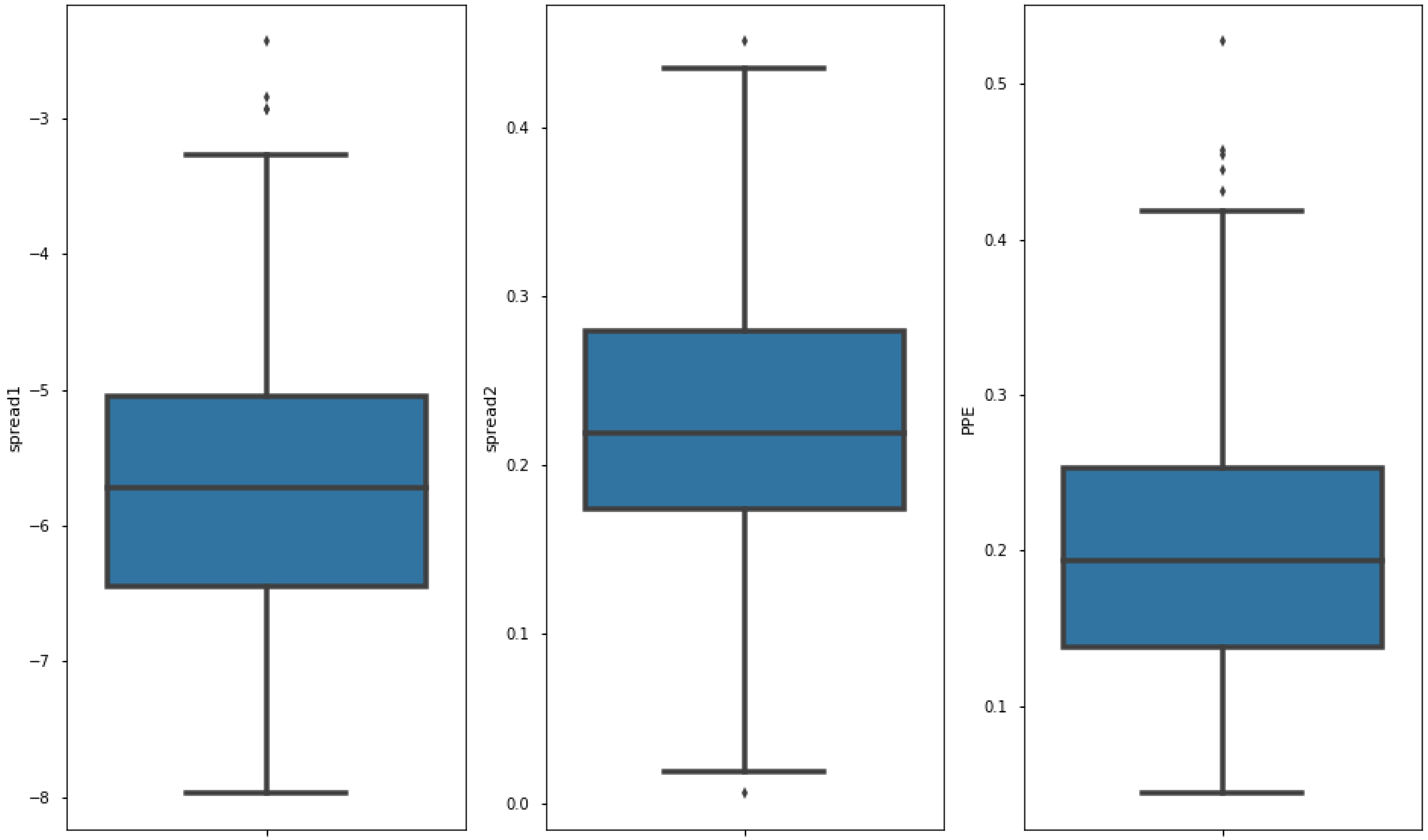
In [76]:
```
df.status.value_counts()
```

Out[76]:
```
1    147
0     48
Name: status, dtype: int64
```

In [77]:
```
# so finally we concluded there are 147 people affected with pd...
```

```
In [92]: fig, ax = plt.subplots(1,3,figsize=(16,10))
         sns.boxplot(x='spread1',data=df, ax=ax[0],orient='v')
         sns.boxplot(x='spread2',data=df, ax=ax[1],orient='v')
         sns.boxplot(x='PPE',data=df,ax=ax[2],orient='v')
```

```
Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x1b384663e80>
```

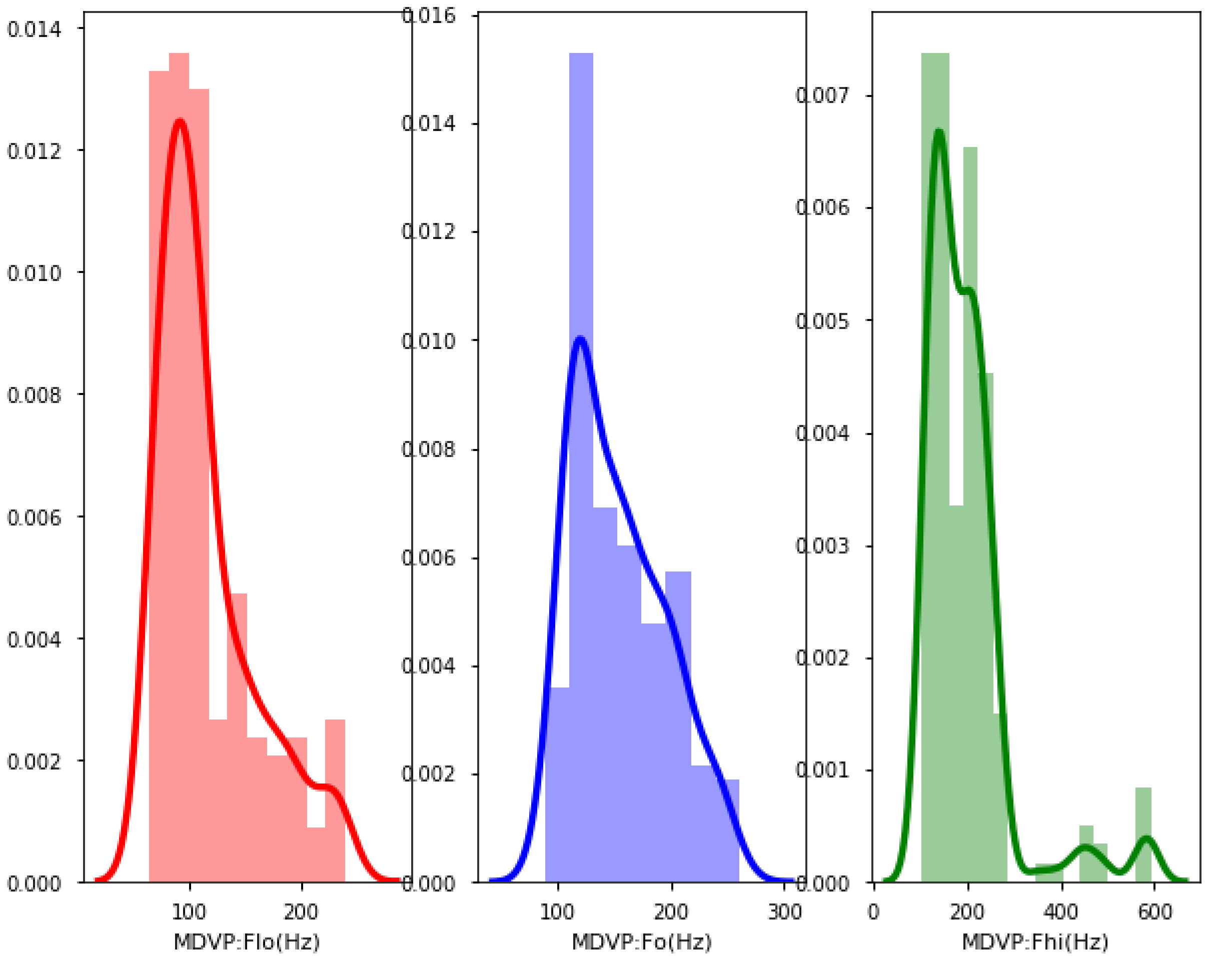


In [79]:

```
fig, ax = plt.subplots(1,3,figsize=(10,8))
sns.distplot(df['MDVP:Flo(Hz)'],ax=ax[0],color="r")
sns.distplot(df['MDVP:Fo(Hz)'],ax=ax[1],color="b")
sns.distplot(df['MDVP:Fhi(Hz)'],ax=ax[2],color="g")
```

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b38310f1d0>
```



In [80]:

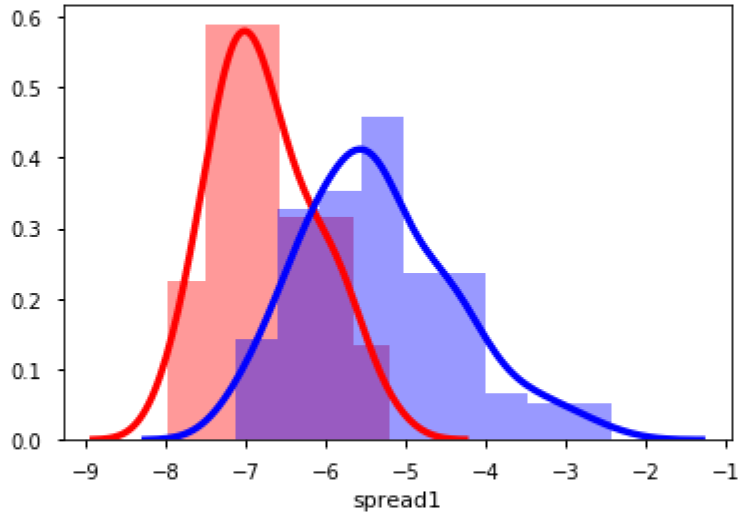
```
# The measures of vocal fundamental frequency are shown above.
#There is a positive skewness for minimum vocal fundemental frequency
# with more high values between 75Hz and 125Hhz. The average vocal
#frequency is almost normally distributed with more values
# ranging 115Hz and 130Hz. The high vocal frequency does not have
#any skewness, but some range of values are at the right most tail
```

In [81]:

```
#Bi Variate analysis:
sns.distplot( df[df.status == 0]['spread1'], color = 'R')
sns.distplot( df[df.status == 1]['spread1'], color = 'B')
```

Out[81]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b383bb8940>
```

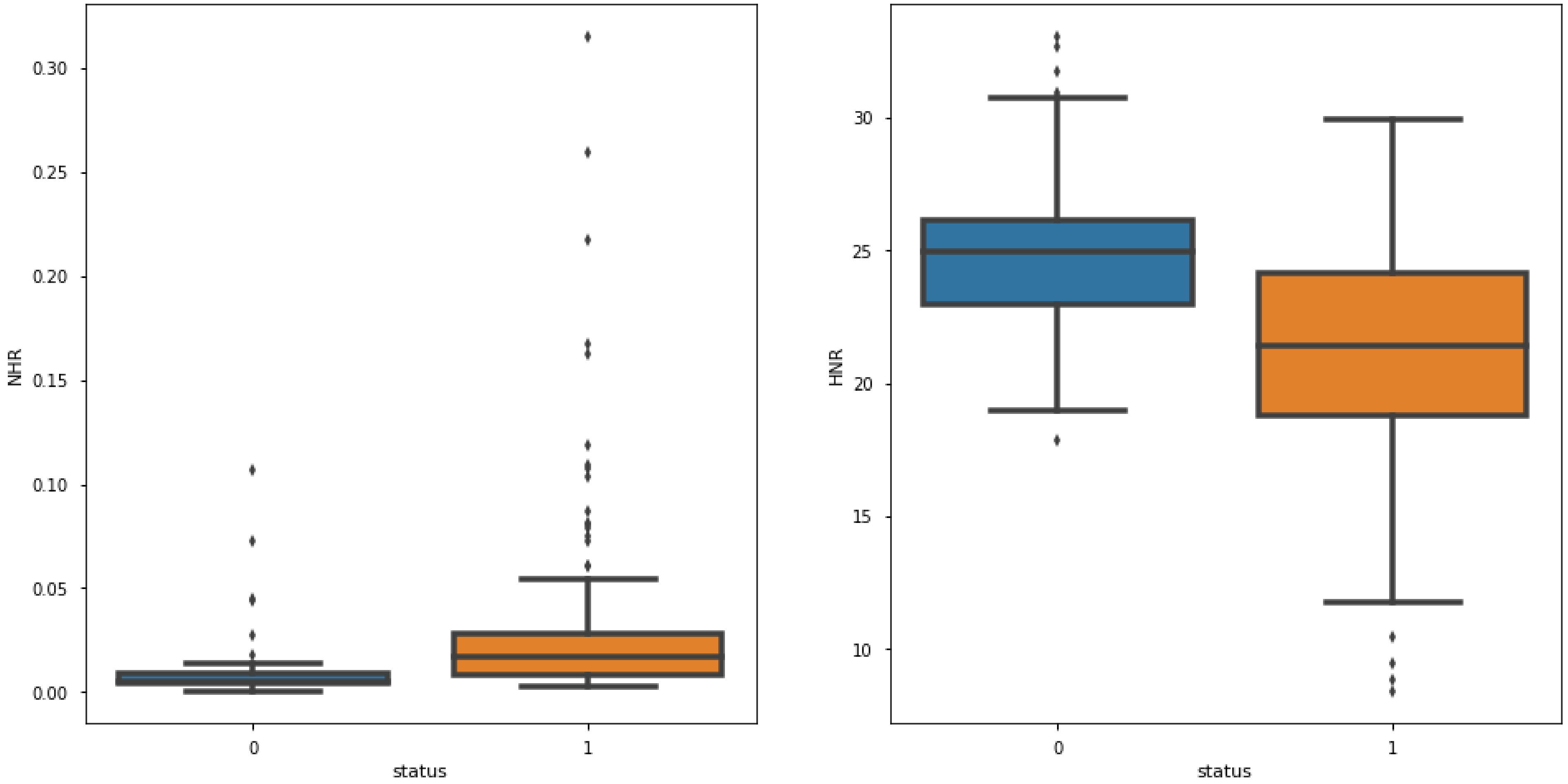


In [82]:

```
# Spread1 is normally distributed between person who have PD and who is normal.
# People who have spread1 between - 8.5 and -7.5 are more and they are normal.
# People whose spread1 is between -6.5 and -5 are having PD
```

```
In [93]: fig, ax = plt.subplots(1,2,figsize=(16,8))
         sns.boxplot(x='status',y='NHR',data=df,ax=ax[0])
         sns.boxplot(x='status',y='HNR',data=df,ax=ax[1])
```

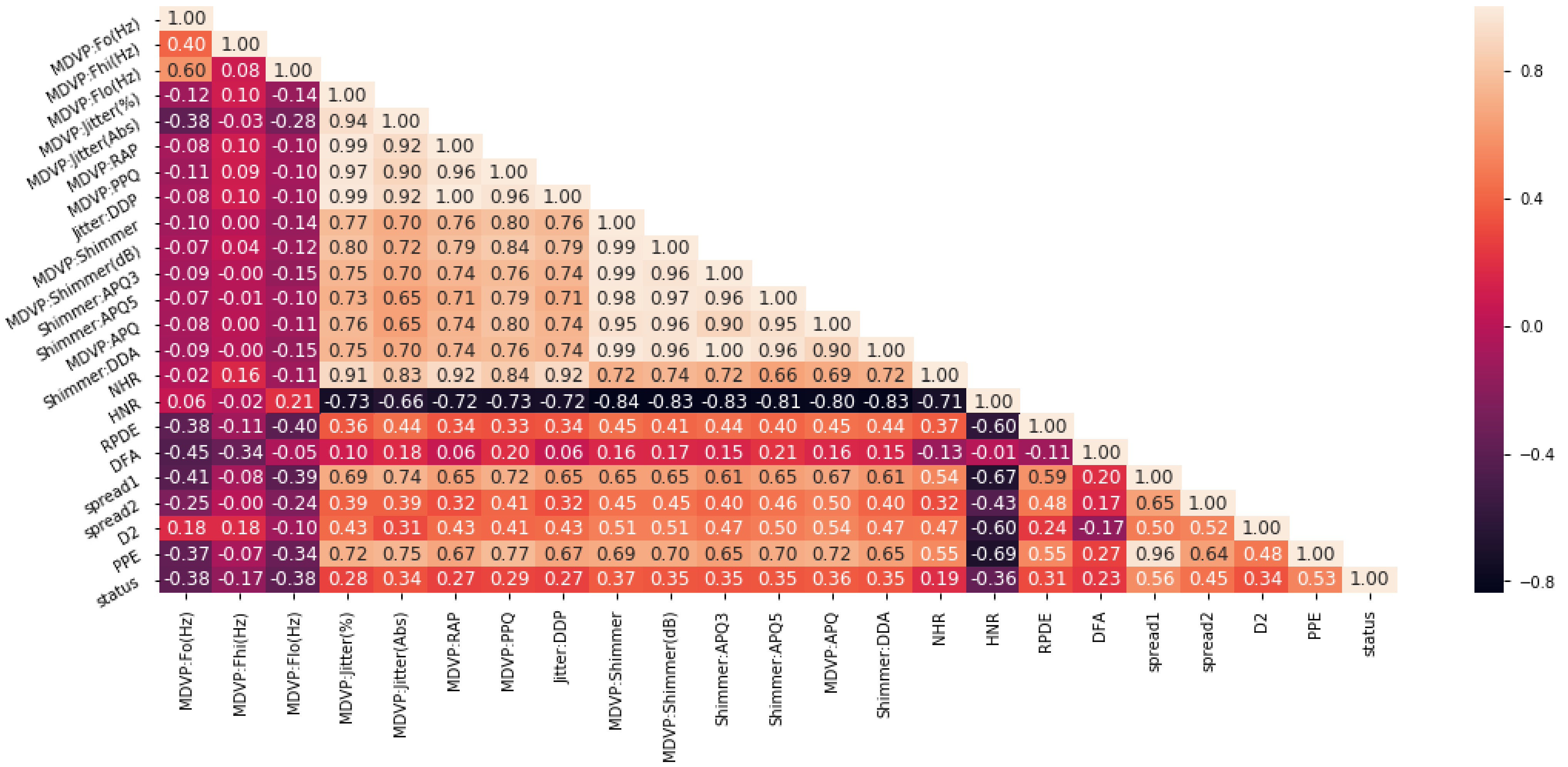
```
Out[93]: <matplotlib.axes._subplots.AxesSubplot at 0x1b3836fd208>
```



```
In [84]: # People who have PD(status equal to one) have higher levels of Noise to Harmonic ratio.
         # Also, looking into the HNR ratio people who have PD have lower levels in the same.
```

In [85]:

```
# Correlation comparision with heat map
corr = df.corr()
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth":
3.5})
plt.figure(figsize=(18,7))
# create a mask so we only see the correlation values once
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True
a = sns.heatmap(corr,mask=mask, annot=True, fmt='.2f')
rotx = a.set_xticklabels(a.get_xticklabels(), rotation=90)
roty = a.set_yticklabels(a.get_yticklabels(), rotation=30)
```



	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	RPDE	DFA	spread1	spread2	D2	PPE	status
MDVP:Fo(Hz)	1.00																						
MDVP:Fhi(Hz)	0.40	1.00																					
MDVP:Flo(Hz)	0.60	0.08	1.00																				
MDVP:Jitter(%)	-0.12	0.10	-0.14	1.00																			
MDVP:Jitter(Abs)	-0.38	-0.03	-0.28	0.94	1.00																		
MDVP:RAP	-0.08	0.10	-0.10	0.99	0.92	1.00																	
MDVP:PPQ	-0.11	0.09	-0.10	0.97	0.90	0.96	1.00																
Jitter:DDP	-0.08	0.10	-0.10	0.99	0.92	1.00	0.96	1.00															
MDVP:Shimmer	-0.10	0.00	-0.14	0.77	0.70	0.76	0.80	0.76	1.00														
MDVP:Shimmer(dB)	-0.07	0.04	-0.12	0.80	0.72	0.79	0.84	0.79	0.99	1.00													
Shimmer:APQ3	-0.09	-0.00	-0.15	0.75	0.70	0.74	0.76	0.74	0.99	0.96	1.00												
Shimmer:APQ5	-0.07	-0.01	-0.10	0.73	0.65	0.71	0.79	0.71	0.98	0.97	0.96	1.00											
MDVP:APQ	-0.08	0.00	-0.11	0.76	0.65	0.74	0.80	0.74	0.95	0.96	0.90	0.95	1.00										
Shimmer:DDA	-0.09	-0.00	-0.15	0.75	0.70	0.74	0.76	0.74	0.99	0.96	1.00	0.96	0.90	1.00									
NHR	-0.02	0.16	-0.11	0.91	0.83	0.92	0.84	0.92	0.72	0.74	0.72	0.66	0.69	0.72	1.00								
HNR	0.06	-0.02	0.21	-0.73	-0.66	-0.72	-0.73	-0.72	-0.84	-0.83	-0.83	-0.81	-0.80	-0.83	-0.71	1.00							
RPDE	-0.38	-0.11	-0.40	0.36	0.44	0.34	0.33	0.34	0.45	0.41	0.44	0.40	0.45	0.44	0.37	-0.60	1.00						
DFA	-0.45	-0.34	-0.05	0.10	0.18	0.06	0.20	0.06	0.16	0.17	0.15	0.21	0.16	0.15	-0.13	-0.01	-0.11	1.00					
spread1	-0.41	-0.08	-0.39	0.69	0.74	0.65	0.72	0.65	0.65	0.65	0.61	0.65	0.67	0.61	0.54	-0.67	0.59	0.20	1.00				
spread2	-0.25	-0.00	-0.24	0.39	0.39	0.32	0.41	0.32	0.45	0.45	0.40	0.46	0.50	0.40	0.32	-0.43	0.48	0.17	0.65	1.00			
D2	0.18	0.18	-0.10	0.43	0.31	0.43	0.41	0.43	0.51	0.51	0.47	0.50	0.54	0.47	0.47	-0.60	0.24	-0.17	0.50	0.52	1.00		
PPE	-0.37	-0.07	-0.34	0.72	0.75	0.67	0.77	0.67	0.69	0.70	0.65	0.70	0.72	0.65	0.55	-0.69	0.55	0.27	0.96	0.64	0.48	1.00	
status	-0.38	-0.17	-0.38	0.28	0.34	0.27	0.29	0.27	0.37	0.35	0.35	0.35	0.36	0.35	0.19	-0.36	0.31	0.23	0.56	0.45	0.34	0.53	1.00

In [86]:

```
# MDVP:Jitter(%) has a very high correlation with MDVP:Jitter(Abs),MDVP:RAP,MDVP:PPQ,Jitter:DDP
# MDVP:Shimmer has a very correlation with
# MDVP:Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,MDVP:APQ,Shimmer:DDA this may be because they are related to each other.
#This may be because multi-dimensinal voice programs analysis is closely related with these variables
#The target variable status has a weak positive corelation with spread1
```

In [87]:

```
# Initialize a MinMaxScaler and scale the features to between -1 and 1 to normalize them.
# The MinMaxScaler transforms features by scaling them to a given range.
# The fit_transform() method fits to the data and then transforms it.
# We don’t need to scale the labels.
```

```
In [88]: scaler=MinMaxScaler((-1,1))
         x=scaler.fit_transform(features)
         y=labels

         C:\Users\YASWANTH\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning: Data with input dtype object was converted
         to float64 by MinMaxScaler.
           warnings.warn(msg, DataConversionWarning)
```

```
In [89]: #spliting of dataset only 20% of testing
         x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2, random_state=7)
```

```
In [90]: # by training the model using the XGB classifier
         model=XGBClassifier()
         model.fit(x_train,y_train)

Out[90]: XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
                colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
                importance_type='gain', interaction_constraints=None,
                learning_rate=0.300000012, max_delta_step=0, max_depth=6,
                min_child_weight=1, missing=nan, monotone_constraints=None,
                n_estimators=100, n_jobs=0, num_parallel_tree=1,
                objective='binary:logistic', random_state=0, reg_alpha=0,
                reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None,
                validate_parameters=False, verbosity=None)
```

```
In [91]: #for accuracy of classification @GURRAM
         y_pred=model.predict(x_test)
         print(accuracy_score(y_test, y_pred)*100)

         94.8717948718
```