17. Parkinson’s project

/* The Parkinson dataset includes 195 biomedical records of people with 23 varied characteristics. The idea behind this project is to design an ML model that can differentiate between healthy people and those suffering from Parkinson’s disease. The model uses the XGboost (extreme gradient boosting) algorithm based on decision trees to make the separation.*/

```
Parkinson's signs and symptoms may include:

1-Tremor. A tremor, or shaking, usually begins in a limb, often your hand or fingers.
2-Slowed movement (bradykinesia). Over time, Parkinson's disease may slow your movement, making simple tasks difficult and time-consuming.
3-Rigid muscles. Muscle stiffness may occur in any part of your body.
4-Impaired posture and balance. Your posture may become stooped, or you may have balance problems as a result of Parkinson's disease.
5-Loss of automatic movements. You may have a decreased ability to perform unconscious movements, including blinking, smiling or swinging your arms when you walk.
6- speech changes and writting changes...
```

```
Classification is a process of categorizing a given set of data into classes, It can be performed on both structured or unstructured data.

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables.

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.
```

```
The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems.
The tremor fluctuation during kinetic and resting task is used as classification features. The support vector machine is used as a classifier and tested with 10-fold cross-validation. This novel feature provides a perfect PD/ET classification with 100% accuracy, sensitivity and specificity.
```

```
XGBoost is an implementation of Gradient Boosted decision trees.
Boosting is an ensemble technique where new models are added to correct the errors made by existing models.
```

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

Execution Speed.
Model Performance.

In [1]:
```
pip install numpy pandas sklearn xgboost
```

```
Requirement already satisfied: numpy in c:\users\personal\anaconda3\lib\site-packages (1.22.2)
Requirement already satisfied: pandas in c:\users\personal\anaconda3\lib\site-packages (1.1.3)
Requirement already satisfied: sklearn in c:\users\personal\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: xgboost in c:\users\personal\anaconda3\lib\site-packages (1.5.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\personal\anaconda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\personal\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: scikit-learn in c:\users\personal\anaconda3\lib\site-packages (from sklearn) (0.23.2)
Requirement already satisfied: scipy in c:\users\personal\anaconda3\lib\site-packages (from xgboost) (1.5.2)
Requirement already satisfied: six>=1.5 in c:\users\personal\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas)
(1.15.0)
Requirement already satisfied: joblib>=0.11 in c:\users\personal\anaconda3\lib\site-packages (from scikit-learn->sklearn) (0.17.
0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\personal\anaconda3\lib\site-packages (from scikit-learn->sklearn)
(2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```python
import numpy as np
import pandas as pd
import os, sys
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [3]:
```python
#DataFlair - Read the data
df=pd.read_csv('parkinsons.data')
df.head()
```

Out[3]:

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

5 rows × 24 columns

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

```
Jitter is defined as the parameter of frequency variation from cycle to cycle, and shimmer relates to the amplitude variation of
the sound wave
```

```
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
```

In [4]:
```
df.tail()
```

Out[4]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shi
190	phon_R01_S50_2	174.188	230.978	94.261	0.00459	0.00003	0.00263	0.00259	0.00790	0.04087	...	
191	phon_R01_S50_3	209.516	253.017	89.488	0.00564	0.00003	0.00331	0.00292	0.00994	0.02751	...	
192	phon_R01_S50_4	174.688	240.005	74.287	0.01360	0.00008	0.00624	0.00564	0.01873	0.02308	...	
193	phon_R01_S50_5	198.764	396.961	74.904	0.00740	0.00004	0.00370	0.00390	0.01109	0.02296	...	
194	phon_R01_S50_6	214.289	260.277	77.973	0.00567	0.00003	0.00295	0.00317	0.00885	0.01884	...	

5 rows × 24 columns

In [5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	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)	..
count	195.000000	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	0.282251	..
std	41.390065	91.491548	43.521413	0.004848	0.000035	0.002968	0.002759	0.008903	0.018857	0.194877	..
min	88.333000	102.145000	65.476000	0.001680	0.000007	0.000680	0.000920	0.002040	0.009540	0.085000	..
25%	117.572000	134.862500	84.291000	0.003460	0.000020	0.001660	0.001860	0.004985	0.016505	0.148500	..
50%	148.790000	175.829000	104.315000	0.004940	0.000030	0.002500	0.002690	0.007490	0.022970	0.221000	..
75%	182.769000	224.205500	140.018500	0.007365	0.000060	0.003835	0.003955	0.011505	0.037885	0.350000	..
max	260.105000	592.030000	239.170000	0.033160	0.000260	0.021440	0.019580	0.064330	0.119080	1.302000	..

8 rows × 23 columns

In [7]:
```
df.shape
```

Out[7]: (195, 24)

In [8]:
```
df.mean()
```

Out[8]:
```
MDVP:Fo(Hz)         154.228641
MDVP:Fhi(Hz)        197.104918
MDVP:Flo(Hz)        116.324631
MDVP:Jitter(%)        0.006220
MDVP:Jitter(Abs)      0.000044
MDVP:RAP              0.003306
MDVP:PPQ              0.003446
Jitter:DDP            0.009920
MDVP:Shimmer          0.029709
MDVP:Shimmer(dB)      0.282251
Shimmer:APQ3          0.015664
Shimmer:APQ5          0.017878
MDVP:APQ              0.024081
Shimmer:DDA           0.046993
NHR                   0.024847
HNR                  21.885974
status                0.753846
RPDE                  0.498536
DFA                   0.718099
spread1              -5.684397
spread2               0.226510
D2                    2.381826
PPE                   0.206552
dtype: float64
```

In [10]:
```
df.columns
```

Out[10]:
```
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 [11]:
```
#DataFlair - Get the features and labels
features=df.loc[:,df.columns!='status'].values[:,1:]
labels=df.loc[:,'status'].values
```

In [12]:
```python
#DataFlair - Get the count of each label (0 and 1) in labels
print(labels[labels==1].shape[0], labels[labels==0].shape[0])
```

```
147 48
```

In [13]:
```python
#DataFlair - Scale the features to between -1 and 1
scaler=MinMaxScaler((-1,1))
x=scaler.fit_transform(features)
y=labels
```

In [14]:
```python
#DataFlair - Split the dataset
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2, random_state=7)
```

In [15]:
```python
#DataFlair - Train the model
model=XGBClassifier()
model.fit(x_train,y_train)
```

```
C:\Users\personal\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=Fa
lse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cl
ass - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[12:00:06] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eva
l_metric if you'd like to restore the old behavior.
```

Out[15]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [16]:
```
# DataFlair - Calculate the accuracy
y_pred=model.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
```

94.87179487179486

In [17]:
```
from sklearn.metrics import confusion_matrix

pd.DataFrame(

    confusion_matrix(y_test, y_pred),

    columns=['Predicted Healthy', 'Predicted Parkinsons'],

    index=['True Healthy', 'True Parkinsons']

)
```

Out[17]:

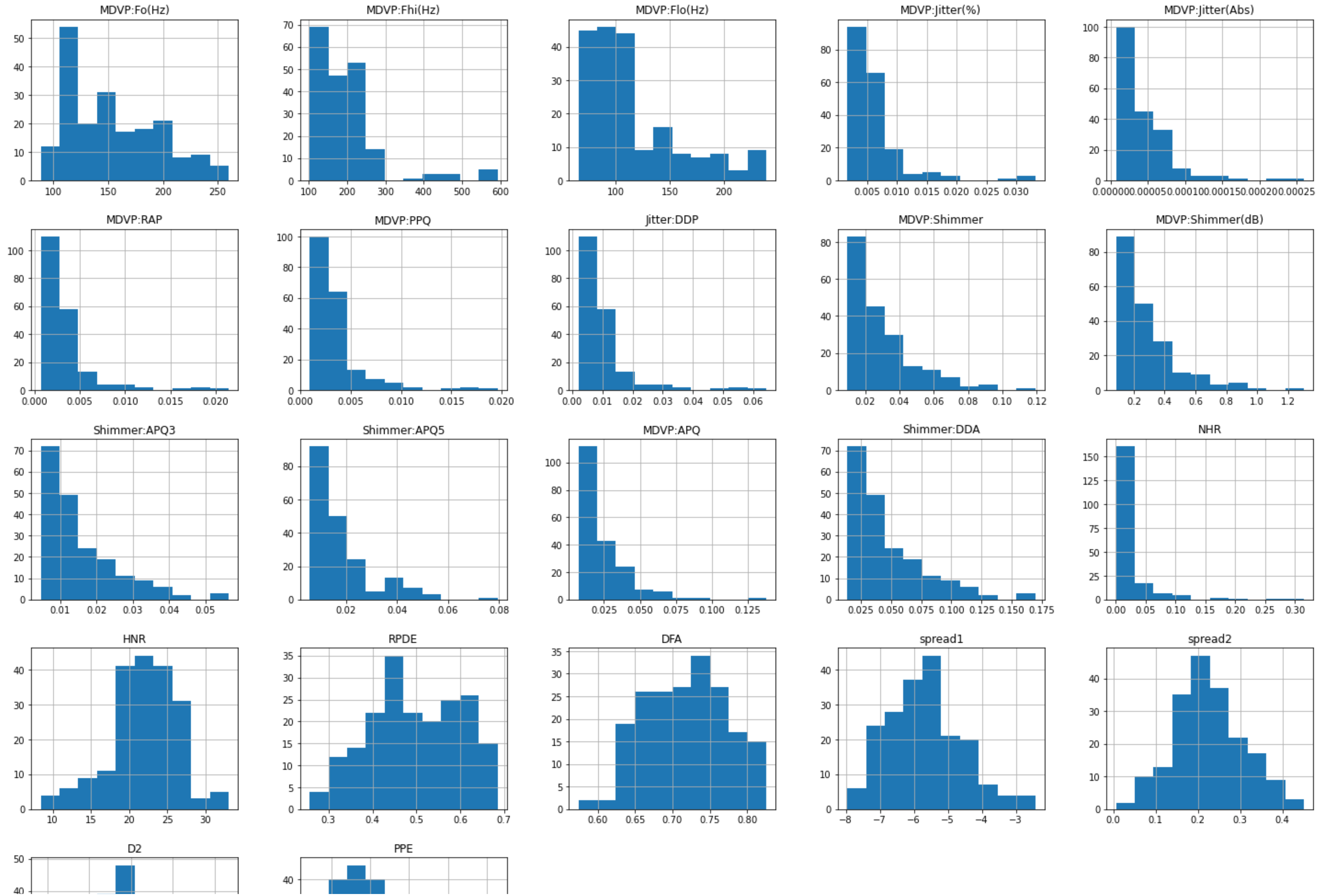
	Predicted Healthy	Predicted Parkinsons
True Healthy	6	1
True Parkinsons	1	31

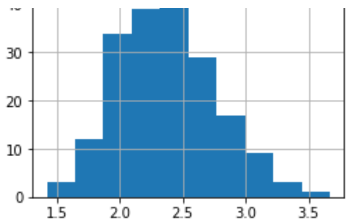
In [18]:
```
#Determining Dependent & Independent Variables
# get features and labels

x=df.loc[:,df.columns!='status'].values[:,1:]
x1=df.loc[:,df.columns!='status']
y=df.loc[:,'status'].values
y1=df.loc[:,'status']
```

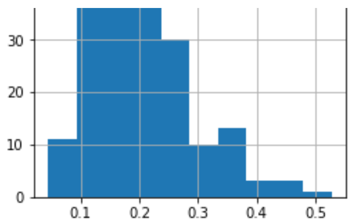
In [24]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [123]:
```python
#Analyzing Features
x1.hist(figsize=(25,20))
plt.show()
```



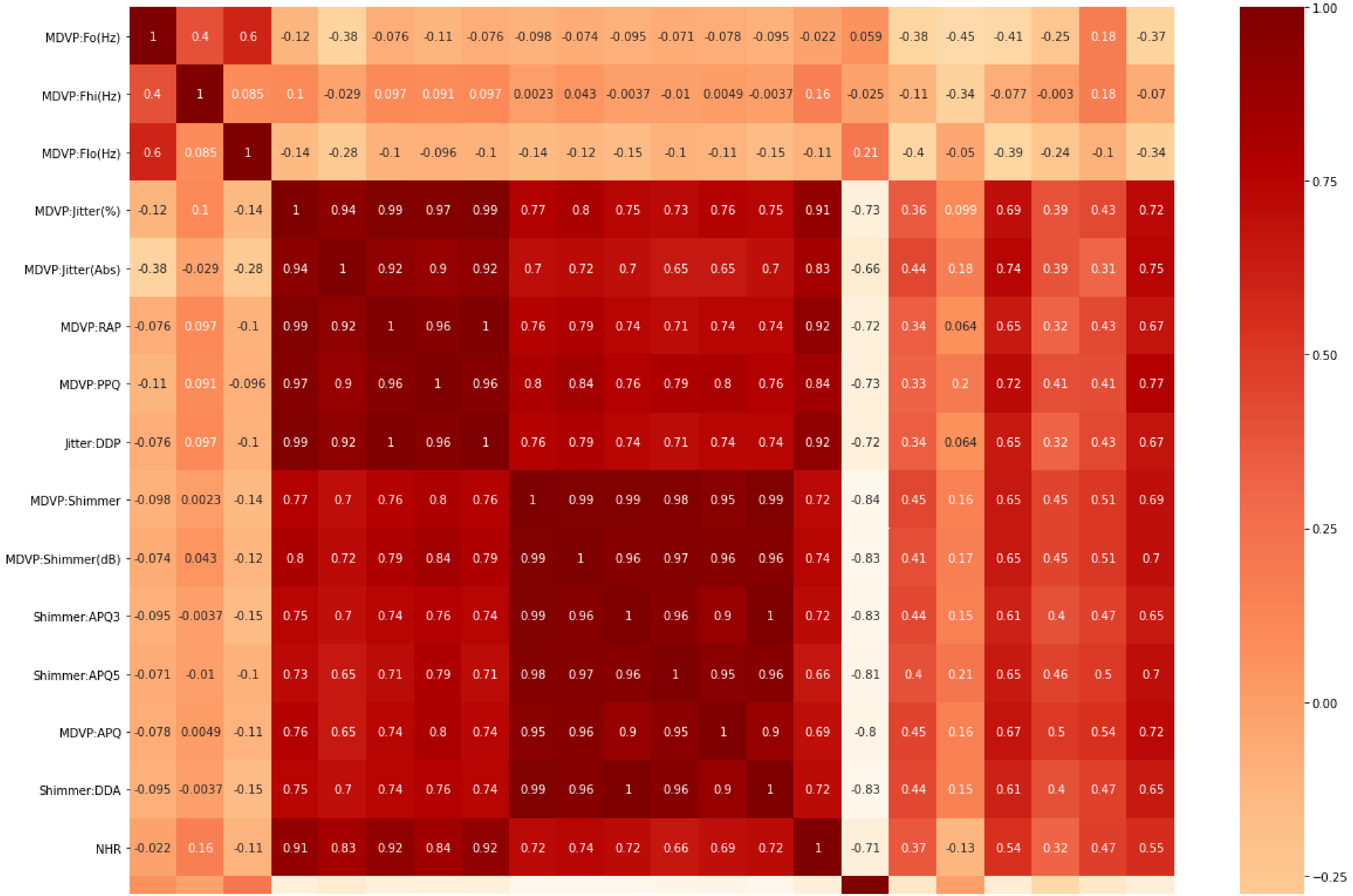


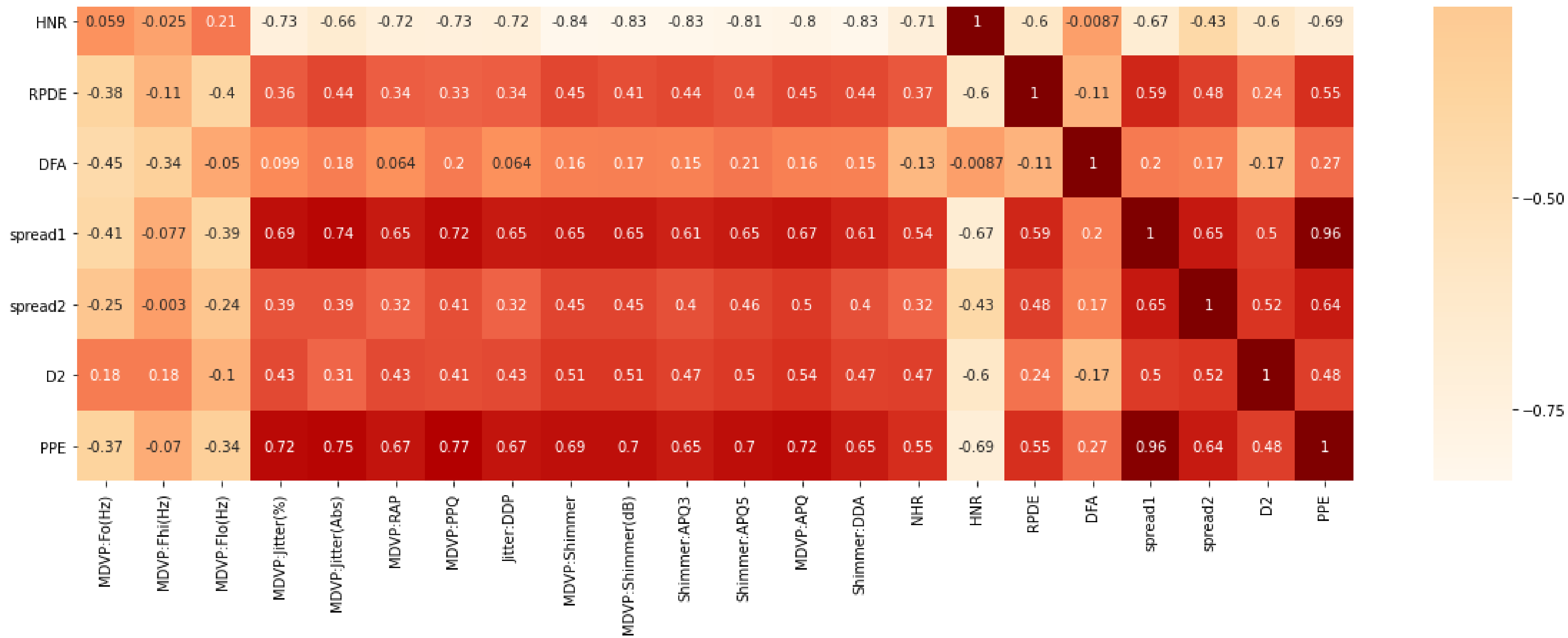
30
20
10
0
1.5
2.0
2.5
3.0
3.5



30
20
10
0
0.1
0.2
0.3
0.4
0.5

In [25]:
```
correl=x1.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correl,annot=True,cmap='OrRd')
plt.show()
```





In [26]:
```
#Scale the features to between -1 and 1
scaler=MinMaxScaler((-1,1))
x1=scaler.fit_transform(x)
y1=y
```

In [27]:
```
#Split the dataset

xtrain,xtest,ytrain,ytest=train_test_split(x1, y1, test_size=0.2)
```

In [28]:
```
# Train the model
from xgboost import XGBClassifier

model=XGBClassifier()
model.fit(xtrain,ytrain)
predict=model.predict(xtest)
```

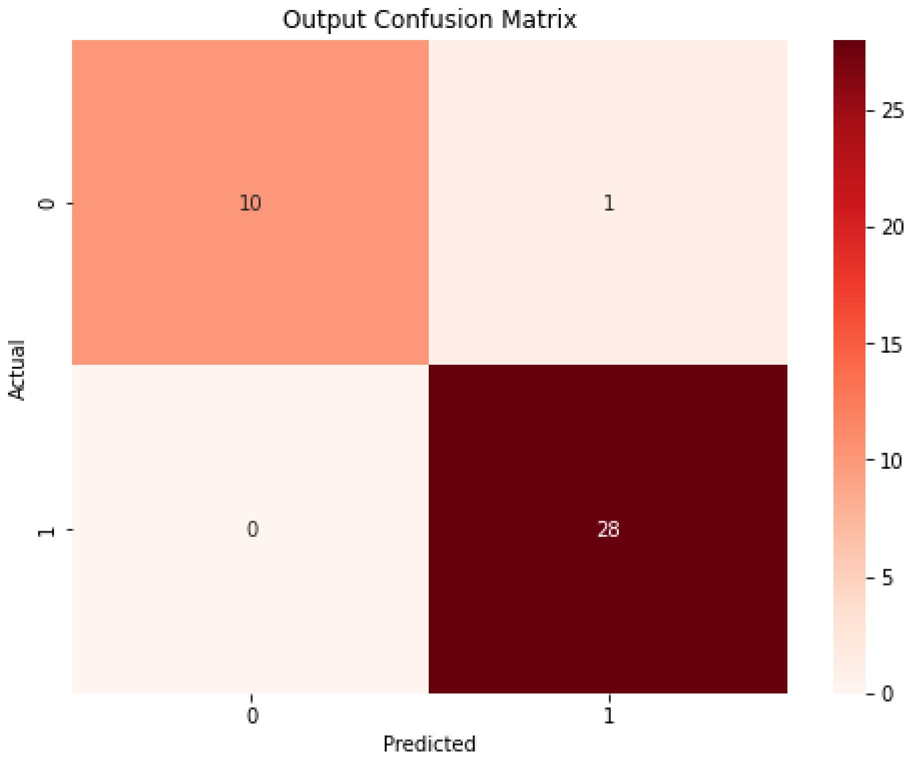
```
[12:04:04] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

In [29]:
```
print(accuracy_score(ytest,predict)*100)
```

```
97.43589743589743
```

In [30]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytest,predict)
plt.figure(figsize=(8,6))
fg=sns.heatmap(cm,annot=True,cmap="Reds")
figure=fg.get_figure()
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Output Confusion Matrix")
```

Out[30]:
```
Text(0.5, 1.0, 'Output Confusion Matrix')
```



In [31]:

```
pd.DataFrame({'actual':ytest,'predict':predict})
```

Out[31]:

	actual	predict
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	0	0
7	0	0
8	0	1
9	1	1
10	1	1
11	1	1
12	1	1
13	0	0
14	1	1
15	1	1
16	1	1
17	0	0
18	1	1
19	0	0
20	1	1
21	0	0
22	1	1
23	1	1
24	0	0
25	0	0

	actual	predict
26	0	0
27	1	1
28	1	1
29	0	0
30	1	1
31	1	1
32	1	1
33	1	1
34	1	1
35	1	1
36	1	1
37	1	1
38	1	1

In [70]:
```
y2_pred = classifi2.predict(xtest)
```

In [71]:
```
#fitting the model in SVM
from sklearn.svm import SVC
classifi2 = SVC()
classifi2.fit(xtrain,ytrain)
print(accuracy_score(ytest, y2_pred)*100)
```

```
28.205128205128204
```

In [53]:
```
#DataFlair - Get the features and labels
feature=df.loc[:,df.columns!='status'].values[:,1:]
label=df.loc[:,'status'].values
```

In [54]:
```
df.columns
```

Out[54]:
```
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 [55]:
```
#DataFlair - Get the count of each label (0 and 1) in labels
print(label[label==1].shape[0], label[label==0].shape[0])
```

```
147 48
```

In [56]:
```
df.shape
```

Out[56]:
```
(195, 24)
```

In [57]:
```
#DataFlair - Scale the features to between -1 and 1
scaler=MinMaxScaler((-1,1))
a=scaler.fit_transform(feature)
b=label
```

In [61]:
```
#DataFlair - Split the dataset
x_train1,x_test1,y_train1,y_test1=train_test_split(x, y, test_size=0.2, random_state=7)
```

In [62]:
```
#DataFlair - Train the model
model=XGBClassifier()
model.fit(x_train1,y_train1)
```

```
[15:13:01] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eva
l_metric if you'd like to restore the old behavior.
```

Out[62]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [64]:
```
# DataFlair - Calculate the accuracy
y_pred1=model.predict(x_test1)
print(accuracy_score(y_test1, y_pred1)*100)
```

```
94.87179487179486
```

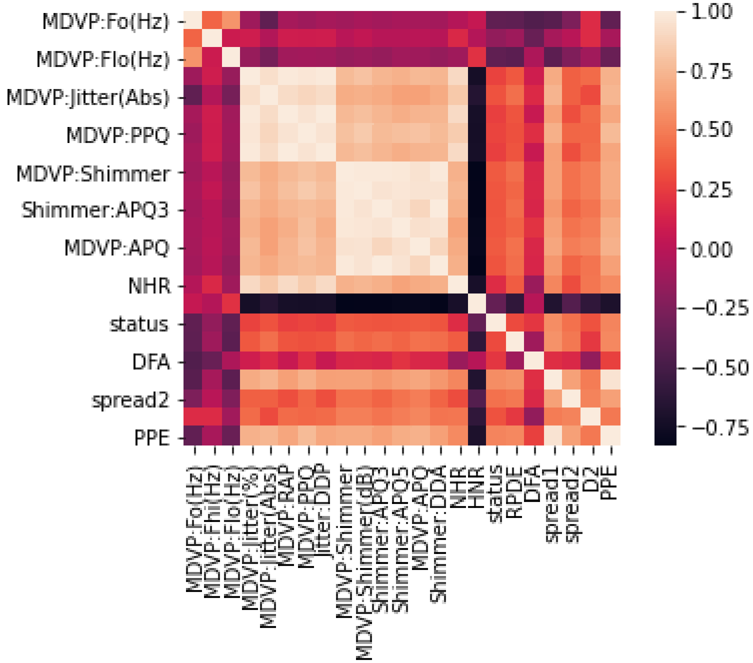
In [66]:
```
from sklearn.svm import SVC
classifi2 = SVC()
```

In [68]:
```
#fitting the model in SVM
classifi2.fit(x_train,y_train)
print(accuracy_score(y_test, y_pred1)*100)
```

```
94.87179487179486
```

In [87]:
```python
import seaborn as sb
corr_map=df.corr()
sb.heatmap(corr_map,square=True)
```

Out[87]: <AxesSubplot:>



In [75]:
```
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 [115]:
```
# split the dataset into input and output attribute.

a1=df['status']
#cols=['MDVP:RAP','Jitter:DDP','DFA','NHR','MDVP:Fhi(Hz)','name','status']
#b1=df.drop(cols,axis=1)
b1=df['spread1']
```

In [116]:
```
# Splitting the dataset into trianing and test set

train_size=0.80
test_size=0.20
seed=5

from sklearn.model_selection import train_test_split
x_tr2,x_te2,y_tr2,y_te2=train_test_split(a1,b1,train_size=train_size,test_size=test_size,random_state=seed)
```

In [117]:
```
#DataFlair - Train the model
model=XGBClassifier()
model.fit(x_tr2,y_tr2)
```

```
[15:59:31] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eva
l_metric if you'd like to restore the old behavior.
```

Out[117]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [118]:
```
# DataFlair - Calculate the accuracy
y_pred2=model.predict(x_te2)
print(accuracy_score(y_te2, y_pred2)*100)
```

```
100.0
```

In [89]:
```
# split the dataset into input and output attribute.

s2=df['status']
cols=['MDVP:RAP','Jitter:DDP','DFA','NHR','MDVP:Fhi(Hz)','name','status']
s1=df.drop(cols,axis=1)
```

In [125]:
```
# Splitting the dataset into trianing and test set

train_size=0.80
test_size=0.20
seed=5

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(s1,s2,train_size=train_size,test_size=test_size,random_state=seed)
```

In [126]:

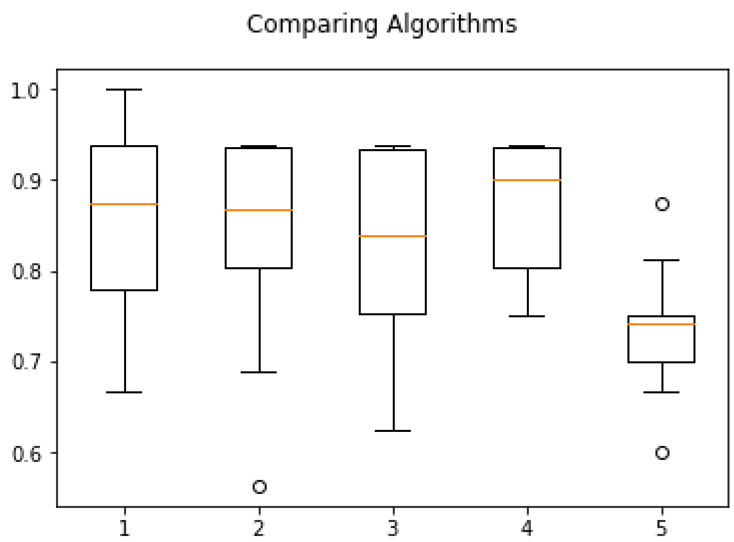
```python
n_neighbors=5
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

# keeping all models in one list
models=[]
models.append(('LogisticRegression',LogisticRegression()))
models.append(('knn',KNeighborsClassifier(n_neighbors=n_neighbors)))
models.append(('SVM',SVC()))
models.append(("XGBOOST",DecisionTreeClassifier()))
models.append(('Naive Bayes',GaussianNB()))

# Evaluating Each model
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
names=[]
predictions=[]
error='accuracy'
for name,model in models:
    fold=KFold(n_splits=10,random_state=0)
    result=cross_val_score(model,x_train,y_train,cv=fold,scoring=error)
    predictions.append(result)
    names.append(name)
    msg="%s : %f (%f)"%(name,result.mean(),result.std())
    print(msg)


# Visualizing the Model accuracy
fig=plt.figure()
fig.suptitle("Comparing Algorithms")
plt.boxplot(predictions)
plt.show()
```

```
LogisticRegression : 0.865833 (0.106275)
knn : 0.834167 (0.118714)
SVM : 0.821667 (0.117951)
XGBOOST : 0.865833 (0.076508)
Naive Bayes : 0.735833 (0.071715)
```



In [127]:

```
#Determining Dependent & Independent Variables
# get features and labels

x=df.loc[:,df.columns!='status'].values[:,1:]
x1=df.loc[:,df.columns!='status']
y=df.loc[:,'status'].values
y1=df.loc[:,'status']
```

In [ ]:

In [ ]: