Parkinson's Disease Detection

In [1]:

```
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 [2]:

```
#Read the data
df=pd.read_csv('/Users/evansabraham/Documents/Jupyter NoteBook/Parkinsons Disease/pa
df.head()
```

Out[2]:

	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	0.00007
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011

5 rows × 24 columns

In [3]:

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

In [4]:

```
#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 [5]:

```
#Scale the features to between -1 and 1
scaler=MinMaxScaler((-1,1))
x=scaler.fit_transform(features)
y=labels
```

In [6]:

```
#Split the dataset
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2, random_state=7)
```

In [7]:

```
#Train the model
model=XGBClassifier()
model.fit(x_train,y_train)
```

Out[7]:

```
▾                                  XGBClassifier
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytr
ee=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='dept
hwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_oneh
ot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_
```

In [8]:

```
#Calculate the accuracy
y_pred=model.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
```

```
94.87179487179486
```