In [10]:

```
# Detecting Parkinson’s Disease with XGBoost
```

In [9]:

```
# Make necessary imports:

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 [ ]:

```
# Now, let’s read the data into a DataFrame and get the first 5 records.
```

In [15]:

```
#DataFlair - Read the data
df=pd.read_csv('C:/Users/KIIT/Downloads/parkinsons.data')
df.head()
```

Out[15]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	M
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	

5 rows × 24 columns

In [16]:

```
# descrive the data

df.describe()
```

Out[16]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Ji
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	19
mean	154.228641	197.104918	116.324631	0.006220	0.000044	0.003306	0.003446	
std	41.390065	91.491548	43.521413	0.004848	0.000035	0.002968	0.002759	
min	88.333000	102.145000	65.476000	0.001680	0.000007	0.000680	0.000920	
25%	117.572000	134.862500	84.291000	0.003460	0.000020	0.001660	0.001860	
50%	148.790000	175.829000	104.315000	0.004940	0.000030	0.002500	0.002690	
75%	182.769000	224.205500	140.018500	0.007365	0.000060	0.003835	0.003955	
max	260.105000	592.030000	239.170000	0.033160	0.000260	0.021440	0.019580	

8 rows × 23 columns

In [17]:

```
#  To know how many rows and cols and NA values

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 [18]:

```
#  shape of the dataset

df.shape
```

Out[18]:

```
(195, 24)
```

In [19]:

```
#  get the all features except "status"

features = df.loc[:, df.columns != 'status'].values[:, 1:] # values use for array format



# get status values in array format

labels = df.loc[:, 'status'].values
```

In [20]:

```
# to know how many values for 1 and how many for 0 labeled status

df['status'].value_counts()
```

Out[20]:

```
1    147
0     48
Name: status, dtype: int64
```

In [21]:

```
#  Initialize MinMax Scaler classs for -1 to 1

scaler = MinMaxScaler((-1, 1))

# fit_transform() method fits to the data and
# then transforms it.

X = scaler.fit_transform(features)
```

```
y = labels

#  Show X and y  here
# print(X, y)
```

In [22]:

```
# split the dataset into training and testing sets with 20% of testings

x_train, x_test, y_train, y_test=train_test_split(X, y, test_size=0.15)
```

In [23]:

```
# Load an XGBClassifier and train the model

from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
```

In [24]:

```
# make a instance and fitting the model

model = XGBClassifier()
model.fit(x_train, y_train) # fit with x and y train
```

Out[24]:

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

In [25]:

```
#  Finnaly pridict the model

y_prediction = model.predict(x_test)

print("Accuracy Score is", accuracy_score(y_test, y_prediction) * 100)
```

```
Accuracy Score is 96.66666666666667
```

In [ ]: