Parkinson's Disease Detector

Package installing and importing

In [2]:

```
!pip install xgboost
```

```
Collecting xgboost
  Using cached xgboost-1.4.2-py3-none-win_amd64.whl (97.8 MB)
Requirement already satisfied: scipy in c:\users\usp\anaconda\lib\site-packages (from xgb
oost) (1.1.0)
Requirement already satisfied: numpy in c:\users\usp\anaconda\lib\site-packages (from xgb
oost) (1.14.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.4.2
```

```
WARNING: You are using pip version 21.1.2; however, version 21.1.3 is available.
You should consider upgrading via the 'c:\users\usp\anaconda\python.exe -m pip install --
upgrade pip' command.
```

In [15]:

```
import numpy as np
import pandas as pd
import os, sys
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Reading Data

In [11]:

```
file = pd.read_csv(r"C:\Users\USP\Desktop\MFE Summer Assignments\June 2021\Parkinson ML\p
arkinsons.data")
file.head()
```

Out[11]:

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

5 rows × 24 columns

Data Stats:

In [12]:

```
file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
name                195 non-null object
MDVP:Fo(Hz)         195 non-null float64
MDVP:Fhi(Hz)        195 non-null float64
MDVP:Flo(Hz)        195 non-null float64
MDVP:Jitter(%)      195 non-null float64
MDVP:Jitter(Abs)    195 non-null float64
MDVP:RAP            195 non-null float64
MDVP:PPQ            195 non-null float64
Jitter:DDP          195 non-null float64
MDVP:Shimmer        195 non-null float64
MDVP:Shimmer(dB)    195 non-null float64
Shimmer:APQ3        195 non-null float64
Shimmer:APQ5        195 non-null float64
MDVP:APQ            195 non-null float64
Shimmer:DDA         195 non-null float64
NHR                 195 non-null float64
HNR                 195 non-null float64
status              195 non-null int64
RPDE                195 non-null float64
DFA                 195 non-null float64
spread1             195 non-null float64
spread2             195 non-null float64
D2                  195 non-null float64
PPE                 195 non-null float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.6+ KB
```

In [13]:

```
file["name"].value_counts().head()
```

Out[13]:

```
phon_R01_S27_6    1
phon_R01_S17_4    1
phon_R01_S20_1    1
phon_R01_S27_4    1
phon_R01_S10_4    1
Name: name, dtype: int64
```

In [14]:

```
file.describe()
```

Out[14]:

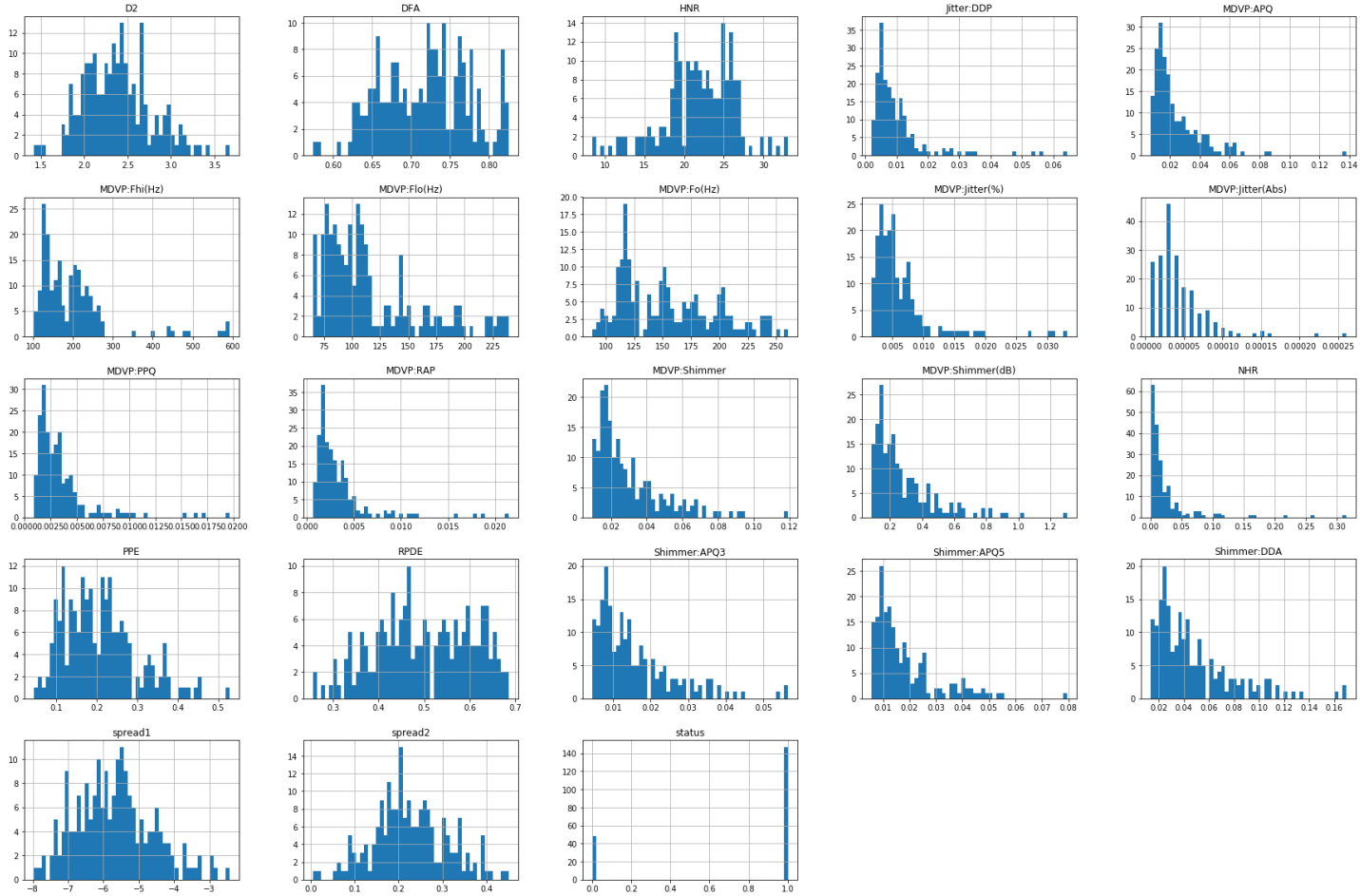
	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDV
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	
mean	154.228641	197.104918	116.324631	0.006220	0.000044	0.003306	0.003446	0.009920	
std	41.390065	91.491548	43.521413	0.004848	0.000035	0.002968	0.002759	0.008903	
min	88.333000	102.145000	65.476000	0.001680	0.000007	0.000680	0.000920	0.002040	
25%	117.572000	134.862500	84.291000	0.003460	0.000020	0.001660	0.001860	0.004985	
50%	148.790000	175.829000	104.315000	0.004940	0.000030	0.002500	0.002690	0.007490	
75%	182.769000	224.205500	140.018500	0.007365	0.000060	0.003835	0.003955	0.011505	
max	260.105000	592.030000	239.170000	0.033160	0.000260	0.021440	0.019580	0.064330	

8 rows × 23 columns

Plotting:

In [16]:

```
file.hist(bins = 50, figsize = (30, 20))
plt.show()
```



The 'status' column has values 0 and 1 as labels; let's get the counts of these labels for both- 0 and 1.

In [5]:

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

In [6]:

```
print(labels[labels==1].shape[0], labels[labels==0].shape[0])
```

```
147 48
```

We have 147 ones and 48 zeros in the status column in our dataset.

Initializing a MinMaxScaler

In [7]:

```
scaler=MinMaxScaler((-1,1))
x=scaler.fit_transform(features)
y=labels
```

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

Splitting Data

Splitting Data

In [8]:

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

Initializing an XGBClassifier

In [9]:

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

```
[13:12:00] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: 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.
```

```
C:\Users\USP\Anaconda\lib\site-packages\xgboost\sklearn.py:1146: 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=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[9]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
       importance_type='gain', 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=4, num_parallel_tree=1,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
       use_label_encoder=True, validate_parameters=1, verbosity=None)
```

Accuracy Prediction

In [10]:

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

```
94.87179487179486
```

```
C:\Users\USP\Anaconda\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth value of an empty array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check that an array is not empty.
  if diff:
```

In this Python machine learning project, we learned to detect the presence of Parkinson's Disease in individuals using various factors. We used an XGBClassifier for this and made use of the sklearn library to prepare the dataset. This gives us an accuracy of 94.87%, which is great considering the number of lines of code in this python project.