In [1]:
```
import numpy as np
import pandas as pd

# ok packages
import os, sys
```

Data Collection

In [4]:
```
# let's read the data into a Dataframe

df = pd.read_csv(r"C:\Users\chitt\Downloads\parkinsons.data")
df.tail()  # show the last 5 rows

# head() use for first 5 rows
```

Out[4]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MD
190	phon_R01_S50_2	174.188	230.978	94.261	0.00459	
191	phon_R01_S50_3	209.516	253.017	89.488	0.00564	
192	phon_R01_S50_4	174.688	240.005	74.287	0.01360	
193	phon_R01_S50_5	198.764	396.961	74.904	0.00740	
194	phon_R01_S50_6	214.289	260.277	77.973	0.00567	

5 rows × 24 columns

In [6]:
```
# describe the data

df.describe()
```

Out[6]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)
count	195.000000	195.000000	195.000000	195.000000	195.000000
mean	154.228641	197.104918	116.324631	0.006220	0.000044
std	41.390065	91.491548	43.521413	0.004848	0.000035
min	88.333000	102.145000	65.476000	0.001680	0.000007
25%	117.572000	134.862500	84.291000	0.003460	0.000020
50%	148.790000	175.829000	104.315000	0.004940	0.000030
75%	182.769000	224.205500	140.018500	0.007365	0.000060
max	260.105000	592.030000	239.170000	0.033160	0.000260

8 rows × 23 columns

In [8]:
```
# To know how man rows and col 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 [10]:
```
# shape of the dataset
df.shape
```

Out[10]: (195, 24)

Feature Enginiearing

In [13]:
```
# get the all features except "status"

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



# get status values in array format

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

In [15]:
```
# to know how many values for 1 and how many for 0 labeled status

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

Out[15]:
```
status
1    147
0     48
Name: count, dtype: int64
```

In [19]:
```
# import MinMaxScaler class from sklearn.preprocessing

from sklearn.preprocessing import MinMaxScaler
```

In [21]:
```
# Initialize MinMax Scaler classes for -1 to 1

scaler = MinMaxScaler((-1, 1))

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

X = scaler.fit_transform(features)
y = labels

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

In [23]:
```
# import train_test_split from sklearn.

from sklearn.model_selection import train_test_split
```

In [27]:
```
# 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)
```

Model Training

In [30]:
```
# Load an XGBClassifier and train the model

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

In [34]:
```
# make a instace and fitting the model

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

Out[34]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rou
nds=None,
              enable_categorical=False, eval_metric=None, feature_ty
pes=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_
bin=None,
```

Model Prediction

In [37]:
```
# Finally pridict the model
y_prediction = model.predict(X_test)

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

```
Accuracy Score is 90.0
```

Summary

** 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 96.66%, which is great considering the number of lines of code in this python project.Hope you enjoyed this Python project. We have already provided you the links for more interesting Python Projects at the top of the blog.**

Completed

In [ ]:
```

```