```
#  install necessary packages ( install first time only )
# !pip install numpy pandas sklearn xgboost --upgrade
```

- Install Necessary packages here

Data Collection

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from imblearn.over_sampling import RandomOverSampler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from tqdm.notebook import tqdm
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings('ignore')
```

```
#  let’s read the data into a DataFrame

df = pd.read_csv('/content/parkinsons.data')
df.tail() # shows the last 5 rows

# head() <= Use for first 5 rows
```

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

5 rows × 24 columns

```
# descrive the data

df.describe()
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimme
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.00000

```
#  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
```

- we can see here there are 135 records and 24 columns available in this dataset

```
#  shape of the dataset

df.shape
```

```
(195, 24)
```

Feature Enginiearing

```
#  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



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

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

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

```
#  import MinMaxScaler class from sklearn.preprocessing

from sklearn.preprocessing import MinMaxScaler



#  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)


#  import train_test_split from sklearn.

from sklearn.model_selection import train_test_split


# 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

```
# Load an XGBClassifier and train the model

from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score


# make a instance and fitting the model

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

```
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, ...)
```

Model Prediction

```
#  Finnaly pridict the model

y_prediction = model.predict(x_test)

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

```
Accuracy Score is 83.33333333333334
```

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.